

# NORTHERN BYTES



## Volume 6 Number 4

Greetings! Summer is finally here again and I am starting to get caught up with the backlog of mail, articles, etc. from last winter. If you sent me something and haven't yet seen it in NORTHERN BYTES and think that I forgot about it - well, you may be right (I hate to say it, but I'm not the world's most organized person!). Please feel free to drop me a postcard and say "Whatever happened to the article I sent you on ....?" If I have it, I'll dig it out of the pile and try to get it into an upcoming issue, otherwise I'll drop you a card back and let you know that I never received it.

It's not that I don't appreciate the articles you send, but this last winter was unreal. It wasn't just articles, of course, but there was plenty to keep me busy. At times I thought I was going to need a bigger mailbox! So, please forgive me if your article got lost in the shuffle, and please drop me a note if you sent something and haven't seen it yet.

This issue is sort of a "catch up" issue, and contains many articles that I've had around for a while. Because it is being prepared shortly after the previous issue, some of our regular features are shorter than usual or absent altogether. Hopefully these features will return next issue.

I mentioned last issue that we were looking at the idea of putting a BBS type operation for TRS-80 users on a packet network. We've examined various options but haven't found anything we could offer at a super good price. So, if you were hoping this would come to pass, please drop me a line immediately. Unless I hear from several people, I won't go any further with this idea. If I do hear from a number of readers, I will send you a mailing describing the options we have, and let you have some input as to which system we go online with.

The future of NORTHERN BYTES is very much in question. As I have mentioned, we're trying one last publicity fling by running a classified ad in Computer Shopper. Unless I get quite a few responses, the next issue may be the last. If you want to see NORTHERN BYTES continue, get your TRS-80-owning friends that have a VISA or Mastercard to send in their card numbers. Remember, this is safe - we do not bill your account until we have actually mailed your copy of NORTHERN BYTES. The cost is \$2.00 per issue (\$3.00 if you live outside North America and want airmail delivery - you must specify that you want airmail, otherwise we assume surface mail is okay). We will continue to mail you each new issue (and bill your credit card) until you tell us not to, your card expires, or we quit publishing, whichever comes first.

I made a couple of statements last issue that could have been a bit misleading in regard to our circulation. So, here's the straight story! We print about 1500 copies of each issue of NORTHERN BYTES, and eventually these are all distributed. However, we have an actual mailing list of about 350. Of those, somewhere between 50 and 100 are folks who have sent us their card numbers as mentioned above. The rest of the 350 are clubs and user groups, folks who have issues coming because they contributed an article or because they were subscribers to Opinion-80 (only a few of those left!), and a few others who get copies for various reasons. The remainder of the 1500 are issues sent free with orders from The Alternate Source, and issues purchased from TAS. Now, as you can see, that means that there are probably about half a million TRS-80 owners out there that don't even know we exist, so tell somebody. We really need at least 2,000 paid (credit card) readers to be able to continue. This would allow the editor to make some money (which would be a brand new experience!) plus it would allow us to eventually get some better equipment (including a Near-Letter Quality printer, which we sorely need).

Note that if we do get two or three thousand credit card subscribers, we will consider that a mandate and will start taking real "subscriptions". Until then, the credit card method is the only way we have to assure you of getting each new issue as it is published. So, the question arises, what do you do if you don't have a credit card? Well, Charley Butler at The Alternate Source (see their ad at the back of this newsletter for their address) has indicated some willingness to consider sending issues on an "open account" type basis. In other words, he sends you an issue, you

send him \$2.00. If you don't send the \$2.00, you don't get the next issue. This is a lot of hassle and we'd prefer not to use this method, but if you don't have a credit card drop Charley a line and maybe he'll do it for you on an individual basis.

Occasionally someone will send, say, \$6.00 and say "Please send the next three issues." What sometimes happens then (depending on who is processing the mail and which address the order is sent to) is that we keep the money and put that person on the mailing list for the three issues, BUT we do not keep any record that money was sent (it appears in our records as if you were an Opinion-80 subscriber with three issues of subscription fulfillment remaining). So, in effect, that person is gambling that we will publish three more issues. If we were to discontinue publication, we would NOT be able to issue a refund since we would have no record that money was sent (and besides, the money would probably have already been spent!). That is why we keep saying that we do NOT take subscriptions. If you have sent money this way in the past, you are hereby advised that you have been playing "Northern Bytes Roulette" and may want to consider switching to the credit card method.

By the way, when ordering copies of Northern Bytes (current or back issues), do not make checks out to "Northern Bytes!" As far as the banks are concerned, we don't exist, so it's an added hassle if you make a check out to us. If you're ordering from The Alternate Source, make the check out to them, not to NORTHERN BYTES.

These opening paragraphs are usually my reflections on various things, and in many cases serve as a letter to all of you since I don't have the time to answer all my mail personally. If you have a question of general interest to Northern Bytes readers that you'd like to see me discuss here, why not drop me a postcard? I'm open to ideas. In the meantime, I hope all of you have a nice summer. I plan to stay away from my computer as much as possible this summer, so if you call me and I say "What's a TRS-80?", you'll know I've been out in the sun too long. Don't worry, summers aren't that long up here - I'll get back to reality sooner or later!

### NEWDOS/80 TIME/DATE PATCH by Paul Fransen

In Northern Bytes Volume 5 Number 4 there is a patch for the date. Instead of the MM/DD/YY format you can use the DD/MM/YY format. This patch fails if the time jumps from 23:59:59 to 00:00:00. If that happens, the month will be incremented instead of the day.

As you probably know, the date you input will be checked (month<13, day<32 and year<99). This is done by comparing with control bytes. The patch mentioned above only disregards the check. A better way is to change the control bytes, so the check can still be done.

The problem can be solved so that the day will be incremented instead of the month, as follows:

The time and date values in memory are in the sequence: seconds, minutes, hours, years, days and months. When the time is updated that sequence is followed. If the time becomes 00:00:00 then the day will be incremented. So the program jumps over the years. We now have changed places (day and month), so there should be an extra jump. It will cost you only one byte, but there is no room left in the program. So, I have searched and found a few free bytes. The bytes can be found at 4CFC. Before you install this patch you had better check to make sure that space is not used by an earlier patch. If it is, then look for other free space.

The patch now is:

SYS0/SYS, sector 01, byte D5: 23 34 C9 becomes C3 FC 4C  
(instead of incrementing the month, you will jump to 4CFC)

SYS0/SYS, sector 10, byte 0D: 00 00 00 00 becomes 23 23 34 C9  
(the day is incremented and then a jump back)



SYS0/SYS, sector 13, byte 88: 4D 4D becomes 44 44  
SYS0/SYS, sector 13, byte 8B: 44 44 becomes 4D 4D  
(Text "MM/DD" becomes "DD/MM")

SYS0/SYS, sector 13, byte BA: 0C becomes 1F  
SYS0/SYS, sector 13, byte BC: 1F becomes 0C  
(Control bytes are exchanged)

#### LETTERS DEPARTMENT

Reminder: Persons sending letters intended for publication should send them on magnetic media or via MCI Mail (especially if longer than a couple of paragraphs). If you are NOT using Allwrite (or Newscript) and your word processor offers the option to save your file in ASCII format, please do so (especially if using SuperScript!). Your cooperation in this matter will help us to bring you a better newsletter!

Dear Jack,

I've been meaning to write for some time and tell you how much I appreciate receiving Northern Bytes. Volume 6, Number 1 arrived today, and I'd like to thank you for the kind words about my 80 Micro article.

I also thought I might contribute something to the fray—both an opinion and some information.

First, the opinion: You said in an answer to Jim Smith's letter, "I don't use TRSDOS 6 any more than I can possibly help it." That's too bad, because I think that TRSDOS 6, and especially version 6.2, is without a doubt the best operating system available on any Z-80 system today (and also far better than PC/MS-DOS, but that's another story). If you or any readers don't have version 6.2, RUN, don't walk, to your local RSCC and demand a copy including the new documentation.

Some support for the opinion! first, I have used every DOS that I know of for the Models I/III/4 except TRSDOS 2.7 and an extinct thing called DoubleDOS. I have found no system that is easier to program in, that will do more at a user level, that has as much flexibility and power, as TRSDOS 6.2.

Now before every grabs a poisoned pen to argue vehemently against my praise of TRSDOS 6.2, please realize that I am talking about the DOS, not BASIC, though with the Alternate Basic as well as LSI's BEEP or Micro-System's 6.x PLUS, I prefer 6.2 for BASIC programming as well.

Also, please realize that this comes from someone who was a dyed-in-the-wool NEWDOS 80 fan for a long time. Apparat is still the only company that has realized (on a TRS-80) that files can be longer than a sector, and that there are other possibilities than sequential and "random" access. But Apparat never figured out device independence nor a decent method for programmers to communicate with DOS routines, and their BASIC overlay system is just short of Byzantine.

Anyway, if you decide to print my opinion of TRSDOS 6.2, please ask folks not to write me to argue. I get too much mail as it is, anyway.

Now for the information. Jim Smith asked for an explanation of the TRSDOS device commands: SET, ROUTE, LINK, and FILTER. These commands are more clearly presented in the new 6.2 documentation, but still could use some further explanation. My description will be from both a system and a user's point of view, since I've always believed that understanding how something works is the best way to getting the most out of a system.

The first concept is that absolutely all byte (or character) I/O is completely device-independent under 6.2. The keyboard, the screen, printer port, the RS-232 port, disk files (if you wish—they can also be set up for record I/O), and anything else you can think of are completely independent. The structure of the DOS is such that each device sends and receives bytes in a standard way with standard register configurations.

The importance of that independence is that no device needs to know what other device it is talking to. Want to type directly to a file? No problem. Want to send printer output to your modem? No problem. Want the input from a plotter to go directly to disk file in Memdisk? Again, no problem.

Now to handle all this there are two closely-related data structures: the device control block (DCB) and file control block (FCB). Each is 8 bytes long, and the only difference between the two is a single bit. Any DCB can be set to allow calls for input, output, and/or control sequences. Each DCB is known to the system (and to the user) by a two-character name (which is preceded by an asterisk when TRSDOS and a user talk about them).

Since only a device can be concerned with byte I/O according to the rules of TRSDOS, anytime we want to create a filter or a new device driver, it first must be associated with a device name. The SET command establishes a new device name in the table of DCBs, and links that device to a /FLT or /DVR program (which are nothing more than /CMD files with a few extra bytes of initialization).

After the DCB has been established and the program moved either to the low-memory driver area or to protected high memory, it must still be attached somehow to the rest of the system. That's where ROUTE, LINK, FILTER and RESET come in. By changing a few bits in each affected DCB, they redirect the flow of byte I/O (and control) requests.

ROUTE simply redirects all I/O from one device to another device. For example, ROUTE \*PR \*DO sends all printer requests to the screen. The connection stays in place until the first device is RESET.

LINK sends I/O directed to the first device to the second one as well. Again, an example: LINK \*DO \*DU will send everything that would normally go to the screen to device \*DU (whatever that has been SET to) as well.

FILTER is very much like ROUTE — everything that would normally go to the first device is sent to the second instead — except that the system inserts the second device into the I/O path ahead of the first device, and expects that new device to communicate with the second (there are fundamental differences here for a programmer, not a user).

What's interesting about all of this is that no device needs to know ahead of time what other device it is being substituted for or what type of I/O path it is filtering. For example, it is quite easy to set SET \*CF CLICK/FLT and then filter the keyboard, the printer, the display, or the RS-232 port with CLICK. Why would anyone want to? Well, I've filtered RS-232 input with CLICK when I was waiting for someone to show up in a CompuServe conference area. As soon as that person does show up, the computer starts clicking and I can return from whatever else I'm doing to talk to them (I usually stop for a second to RESET the RS-232 line first).

There is so much power in these device commands, however, that one must exercise some caution using them. It is quite possible (and very frustrating) to set up a complex series of ROUTEs, LINKs, and FILTERs that wind up as an infinite loop. That's what the DEVICE command is for: it gives a concise description of what everything is doing. Also, note that all these commands are available from BASIC with the SYSTEM command. In several programs, I've included a prompt that asks a user if a report should be sent to the display or printer. If the user wants a printed report, I simply LINK the display and printer in a single command (or ROUTE the display to the printer) instead of having to hassle writing both PRINT and LPRINT statements. Another solution would be to "OPEN" a buffer to device instead of a disk file, and then use PRINT# to send data to that device.

Three other comments about TRSDOS, and what folks will see soon in Micro 80. In my May column, I show how to redesign the user interface. It is easy to add commands, change the way a user sees the system, filter out entire commands, etc. In other words, TRSDOS 6 supports extensive use of shells, and it's a shame that no one has created an extensive series of user shells.

Second, the Model 4 can do windows. In my upcoming June, July, and August columns, I demonstrate a complete window-handler, that allows up to fifteen overlapping windows of any size and restores both the previous screen display and cursor position when window is closed.

I haven't figured out any practical method of multi-tasking on a Model 4 yet (after all, 64K of workspace and a 4 meg clock on a Z-80 are somewhat restrictive), but then I haven't seen really useful multitasking on anything short of a 68000 chip or a VAX anyway.

Last comment: some people have objected to pay to upgrade to 6.2. How much did Apparat charge to change from NEWDOS 2.1 to 80? From 80v1 to 80v2? How much does IBM charge for any DOS at all? Are upgrades from PC/MS-DOS version 1 to version 2 free? How much did Apple users pay to update to PRO-DOS? (only useful with a hard disk, since pathnames on a 128K floppy are superfluous). The new documentation for 6.2 is worth the \$20 that registered owners are asked to pay, and the DOS is much better and faster, though you must turn off SMOOTH if you want reliable type-ahead.

Sorry this is so long. You certainly have my permission to publish any, all, or none in Northern Bytes as you see fit. But give 6.2 a serious try—I didn't like 6.0 at first, but I now avoid everything else as much as possible.

Sincerely,  
Hardin Brothers

[Just in case any of our readers are so out of touch with the TRS-80 world that they don't recognize the name, Hardin Brothers writes The Next Step column for 80-Micro (and has authored many other useful articles as well). Thanks for all the nice comments, Hardin, and we'll be looking forward to your window-handler series!]

By the way, my dislike of TRSDOS 6 is in many ways related to my dislike of LDOS (the father of TRSDOS 6), which I found was a simply awful DOS to try and use on a Model I (that's my opinion, I know others would disagree). But also it has to do with the attitude of some (and I emphasize some) of the people at Logical Systems, Incorporated toward programmers. The attitude I'm referring to is this: "If you are a programmer and write a program and it doesn't work with LDOS, the fault is in your program (EVEN IF IT WORKS FINE UNDER EVERY OTHER DOS) because, after all, LDOS is the only "real" DOS for the TRS-80. You must rewrite your program to work with LDOS."

Under TRSDOS 6 this attitude might be expressed as: "You must NOT directly interact with the DOS or the computer hardware (keyboard, video, printer) in any way other than by using the SVC calls we have provided. For example, if you have a program that requires the user to press a certain combination of keys and our keyboard driver does not decode that combination properly, you will change your program. You must not attempt to directly access the keyboard." Now I can understand that things like ROUTE, LINK, and FILTER won't work properly if you go around bypassing the Device Control Blocks. But in that case you should be able to use an alternate keyboard driver that will provide the needed key combinations, and should not be arbitrarily limited to the combinations that LSI thinks you should be able to use.

I should point out that the portions of the above paragraphs within quotation marks are my interpretation of LSI's attitude toward programmers, and not a direct quotation by any LSI official. I base my interpretation on various writings in the LSI Journal and on conversations with some of the folks at LSI during an "industry get-together" in the summer of '83.

Finally, I have noticed a tendency of both LDOS and the early versions of TRSDOS 6 to go into "silent death" mode (locking up the computer rather than making a graceful exit to an error routine) under some circumstances, while this does not occur on other DOSes. This was a particular problem with Model I LDOS and one reason I finally decided to avoid using it. Perhaps my experiences were not typical, but after all those early frustrations it's pretty hard for me to be objective about the newer versions of TRSDOS 6, especially when I am quite satisfied with the DOS I'm now using. But isn't it nice that we all have a choice - just as you can buy a Chrysler, Ford, or General Motors automobile, you also have your choice of DOSes for the TRS-80, and that competition is probably the main reason that almost ALL of the TRS-80 DOSes are so much better than CP/M or PC/MS-DOS!

I want to make it very clear that regardless of my personal opinion of LDOS/TRSDOS 6, I am still very happy to get submissions for NORTHERN BYTES from users of those DOSes! It may seem at times as though we are only supporting one or two DOSes, but that's only because the users of the other DOSes don't send us much. Believe me, I do NOT throw articles that have anything good to say about a DOS that I don't happen to prefer into the trash can!]

Jack:

Just a note to tell you how much I appreciate Northern Bytes. It is a super newsletter. I am sending the following code for the Model 4 BASIC because it saved my life once!

On the Model 4 you may "protect" BASIC code by using the ,P option. If you ever need to look at that code, you are out of luck unless you unprotect your listing with the following:

```
BASIC 1.1.0 SYSTEM "MEMORY (ADD=X'72CB',WORD='0000')"  
BASIC 1.0.0 SYSTEM "MEMORY (ADD=X'6247',WORD='0000')"
```

I can't take credit for locating this, I saw it on the LSI SIG on CompuServe. I have used it, and like I said, it saved me many hours after an oops.

Keep up the good work.

Sincerely, William C. Huffman

2444 39th Court, New Port Richey, Florida 33552

P.S. Question! How do you disable password protection and checking under TRSDOS 6?

[Thanks for the zap! I'm sure others will find it useful. As for your question about password protection, I thought maybe I had published that before, but can't seem to locate it if I did. Readers, can you help?]

Dear Jack:

I found the control of stepping rate in the new Model 4 TASMOM very difficult to use. There was no 'smooth' transition from the high speeds (6 and 7) to the low range (which seemed to almost die!). I have patched my version as follows. I now have a smooth slowdown or speedup.

Use the 'Find' command to locate 5F 3D 57 1B. Go BACK three bytes from the located address to see 3A nnnn (the disassembled instruction will be LD A,nnnn). Now, use the Modify H command, starting at the address of the 3A to change TASMOM as follows:

Old	Change to
3A	16
?	00
?	1E
5F	6C
3D	00
57	00
1B	<BREAK>

This disassembles to: LD D,0  
LD E,6CH  
NOP  
NOP  
DEC DE  
etc.

Change the 6CH byte to change the stepping rate if you don't like what you get with my preference.

Best regards, Nate Salisbury  
610 Madam Moore's Lane, New Bern, North Carolina 28560

[Nate says this patch should work with all Model 4 versions of TASMOM. Thanks, Nate!]

#### WHERE AM I ?

by A.J. Hagers, Rotterdam, Holland  
Translated by Paul Fransen

In some books the ROM code at the addresses 0BH and 0CH is 'not used' and so it is not explained. But it is an interesting segment of code. In the LDOS manual it is documented under the system label QWHERE.

A CALL 0BH will give you in HL the address of the instruction following the CALL. It is the RET address of the CALL. So what? In the appendix of the TRSDOS 6.1 manual you will find an interesting example of its use (program F).

Sometimes you might like relocatable code. But the Z-80 instruction set does not have relative calls. With QWHERE it is possible to simulate a relative call. Here's an example of a relative call to a subroutine named SUBR:

```
PUSH HL      ;save...  
PUSH BC      ;the...  
PUSH AF      ;registers  
CALL QWHERE  ;where we are (in HL)  
LD BC,9      ;number of bytes after the call  
ADD HL,BC    ;add HL = RET for relative call  
POP AF       ;restore...  
POP BC       ;registers  
EX (SP),HL   ;RET address to stack (HL=original value)  
JR SUBR      ;jump to subroutine  
...          ;this is where the return of SUBR  
...          ;will lead you to  
  
SUBR         ;start of routine  
...  
RET          ;end of routine  
  
QWHERE POP HL ;address 0BH  
JP (HL)      ;address 0CH
```

This will cost you 15 bytes, but sometimes it will come in handy. A disadvantage is that relative data fields are not possible. You can solve that by using the IX register (and using an index to call the data). But if there is more than one call, then IX will not be a constant.

## SOUP UP GOOD OL' SCRIPSIT

Converting the Grandpa of TRS-80 text processors into a comfortable calculating text editor using **NEWDOC/80** version 2  
by Joachim Kelterbaum  
Frankenstr. 305, 4300 Essen 1, West Germany

Though Scripsit is a very old word processing program, it still is quite useful. I mainly use it for writing short memos, program sources, ASCII- and JCL-files. Recently, a friend of mine asked me for help in his particular application of Scripsit. He processes texts containing quite a few calculations. As these texts do not have fixed formats, there is no way that Visicalc, etc. could be used. While editing the texts with a computer he still needs a pocket calculator to compute the results of calculations done in the text. Seems ridiculous, eh?

I thought of combining the computational capabilities of the BASIC ROM with Scripsit. There should also be a way to pass numbers to the calculating facility without retyping them. Some sort of user keys should be established to pass results back to the text. The routine and zaps supplied below do all this and more. First, a description of how to use this routine so you can figure out whether it's of any use to you!

### Function of the calculating Mode

Directly after entering Scripsit by

DO SPS <Enter>

press @I. This will save the tab-line (second line from the bottom) in case you'll need it again for an orientation. It may be restored at any time by pressing @O.

The calculator mode makes use of the two bottom lines of the screen. These will be overwritten with the following text as soon as the calc-mode is entered:

```
C:      V:      ACCU = 0  USER KEY = 0  DEC. = 2
T:      A0: 0.00      A1: 0.00
```

The fields of this command line have the following purposes:

C: Command field. If you invoke a command from within calc-mode, it will be echoed here.

V: Value-field to place a temporary constant which can be entered directly.

ACCU = 0 indicates that A0 is currently selected. You can also select A1.

USER KEY = 0 indicates that user key 0 is currently selected. You have access to UK0...UK7.

DEC. = 2 means that results of computations will be transferred with 2 places after the decimal point into the user key buffers. The values transferred will be rounded to 2 places. Calculations are performed with 4 places internally. You may select a transfer of 4 places here.

T: Temporary register. Each time you enter the calc-mode the value transferred from the text (right after the cursor) will be displayed here. If the text after the cursor does not contain a numerical value, T: will contain 0.

A0:, A1: are the two accumulators you may select. Results of computations you have invoked are always displayed in the selected accum. After each calculation the result will also be transferred into the buffer of the selected user key.

### Invoking the calculator-mode

@P will transfer the value directly after the cursor position in your text into the T: register. The command lines of the calc-mode will be displayed at the bottom of your screen (if not already there).

@T acts just as above, but it will also move the contents of T: to the selected accum.

@+ acts as @P, but will also add the contents of T: to the selected accum. Analogous functions may be invoked by: @- , @\* , @/ (You need not use the shift key for the selected function: i.e. @ works just like @+ ).

You may invoke additional functions while being in the calc-mode. These will be discussed later.

### Leaving the calculator-mode

<Space> will leave the calc-mode and leaves the cursor position unchanged. You may go on processing your text.

<right arrow> performs a tab after leaving calc-mode.  
<left arrow> positions cursor to start of line after leaving.  
<down arrow> advances 1 line after leaving.  
<up arrow> (analogous).

### Placing results back into your document (user keys)

If you selected user key 3 (at some time in the calc mode), pressing @3 will write the result of the last calculation into your document at the cursor position.

Caution: As many applications prefer alignment of decimal points, the results will be transferred backwards into the text (as you have a fixed number of places after the decimal point this will be quite handy if combined with tabbing).

If you selected other user keys previously, the contents of their buffers remain unchanged and may be recalled at any time. If you recall a user key that has not been defined yet, nothing will happen. You may call @0...@7.

### Commands in calculator mode

Ax selects accumulator Ax x = 0...1  
Ux selects user key x x = 0...7  
T transfer contents of T: register into selected accumulator  
+ , - , \* , / selected accum := selected accum + - \* / T: register  
I Input value into intermediate value register V:  
Terminate input by <Enter>  
V transfer V: register to T: register  
0 transfer A0: to T:  
1 transfer A1: to T:  
! copies A0: to A1:  
R rounds selected accum to 2 places after decimal point.  
2 Results of calculations will be transferred into the selected user key with 2 places after decimal point.  
4 same as above, but transfers 4 decimal places.

Actually, it's more complicated to describe the actions than using them. If you combine the above facilities with defining your own tabs, you'll soon feel comfortable with them.

One hint: For a reason I can't explain, you must not use @T as your first invocation of calc-mode. Use @P instead. The program will get stuck if you do @T as your first action. I have not encountered any other bugs so far.

### Well, how does all this work?

As I mentioned above, I used the calculating facilities of the BASIC interpreter in the ROM. This has the great advantage of not eating up too much RAM for the additional code needed. It has another disadvantage, though, that at first caused some headaches: Scripsit uses its own stack right in a memory region where BASIC performs some of its vital functions: i.e. 41FCH and below.

First, I had to move the stack out of this area. Another minor problem was the honoring of HIMEM so I could protect my additional code. This was taken care of by Zap 003 supplied by Apparat. Another point was the interception of the @-key to get hands on the invocation of the additional functions. Then, the screen driver had to be intercepted, so I could process the user keys. I disabled the initialization of the tab-line, so I could enter Scripsit with a predefined tab-line. One routine kept clearing the bottom line each time a character was entered. This was uncomfortable, so I disabled it. There still are occasions where the calc-mode command lines are overwritten (for example when moving to the top of text with shift-up arrow or when entering commands via <break>). These disturbances are rare, though. You can always restore the calc-mode command lines by entering this mode. Finally, BASIC had to fix up its pointers properly in order to work correctly. That was an easy one: I started the program with a /JCL file (see below).

### Now let's get into detail

Below you find a list of all the Zaps needed to link Scripsit to the calc-mode driver:

Format an empty diskette.

Copy a version of the original SCRIPSIT/LC (version 1.0) to SCRIPSIT/BAK on that disk (use this name!).

Apply the following Zaps to SCRIPSIT/BAK. These Zaps contain ZAP 003 supplied by Apparat (don't panic, if those are applied already).



FRS	rel.	Byte	old	new	Comment
00	64		21	2A	ZAP 003
			FF	49	
			00	40	
			25	00	
			7E	00	
			2F	00	
			77	00	
			AE	00	
			20	00	
			F9	00	
00	B8		CD	00	DON'T CLEAR TAB LINE
			1D	00	
			69	00	
00	C4		F3	00	ZAP 003
00	D5		41	FF	RELOCATE STACK
04	34		41	FF	RELOCATE STACK
07	02		41	FF	REL. STACK
11	77		CD	3A	ZAP 003
			6E	B9	
			7A	7C	
11	FC		C4	32	ZAP 003
			EF	B6	
			5D	7C	
			79	C4	
12	00		32	EF	ZAP 003
			B9	5D	
			7C	00	
12	65		41	FF	REL. STACK
14	9D		21	C3	INTERCEPT USER KEY-
			36	03	REQUEST
			40	F6	
15	7D		3A	C3	KEYBOARD INTERCEPT @
			01	00	
			38	F6	
17	54		41	FF	REL. STACK
19	E5		00	2D	EXIT-ZAP
			00	40	
26	55		E5	C9	DISABLE CLEAR CMD-LINE SBR.
28	CF		41	FF	REL. STACK
30	EF		41	FF	REL. STACK
40	E7		41	FF	REL. STACK

Part 2 of the preparations is easy: Write a /JCL file SPS/JCL with the following contents:

```
BASIC
CMD"5
SPSMOD
```

The last part of the necessary modifications is just a little harder: Enter the program at the end of this article into your EDTASM and save the compiled code as SPSMOD/CMD.

Now you're all set. If you didn't make any mistakes, you invoke your brand new 'old Scripsit' by

```
DO SPS <Enter>
```

If you don't have the patience of entering all this stuff into your computer, ask Jack, if he'll put it into one of the public domain disks to come out.

[Will do, as soon as I can find enough spare time to put nother PD disk together! -Jack]

```
00100 ;
00110 ;
00120 ; SPSMOD/SRC
00130 ;
00140 ;a Driver Program to add calculator-mode capability
```

```
00150 ; to SCRIPSIT/LC
00160 ;
00170 ; VER 1.2 04-03-1985
00180 ;
00190 ;by Joachim Kelterbaum, Frankenstr. 385, 4300 Essen 1
00200 ; W. Germany
00210 ;
00220 ;Comments in this source code rather refer to logical
00230 ;segments than to each single statement. Please, refer
00240 ;to Jack Decker's 'TRS80 Rom Routines Documented' for a
00250 ;detailed explanation of the ROM-calls referenced here
00260 ;(That book is well worth it's weight in gold)!
00270 ;
00280 ;
00290 ;
00300 ORG 7EB3H ;Initialization of a predefined
00310 TABTEL DEFN 0101H ;Tab-Line in Scripsit
00320 DEFN 0101H
00330 DEFN 0101H
00340 DEFN 0001H
00350 DEFN 0H
00360 DEFN 0H
00370 DEFN 0H
00380 DEFN 0H
00390 DEFN 0H
00400 DEFN 0H
00410 DEFN 0H
00420 DEFN 0H
00430 DEFN 0H
00440 DEFN 0H
00450 DEFN 0H
00460 DEFN 0H
00470 ;
00480 ;
00490 ORG 5200H
00500 START LD HL,BUFRT ;Initialize user key
00510 DEC HL ;buffer with 0's
00520 LD B,128
00530 LD A,0
00540 HLT LD (HL),A
00550 INC HL
00560 DJNZ HLT
00570 LD HL,SCRMOD ;update HIDEH value
00580 DEC HL
00590 LD (4049H),HL ;plug into HIDEH
00600 ;
00610 LD HL,TX ;start SCRIPSIT/BAK
00620 JP 4405H
00630 TX DEFN 'SCRIPSIT/BAK'
00640 DEFN 00H
00650 ;
00660 ;
00670 ORG 0F600H
00680 SCRMOD EQU $
00690 KBDINT JP KBDI ;link address KBDINT in Scripsit
00700 UKREQ JP UKRG ; " " UK-request "
00710 ; (see Zap-table)
00720 ;
00730 KBDI LD A,(3B01H);this instruction was overzapped
00740 ;by JP KBDINT in Scripsit. It
00750 ;will be executed here.
00760 AND 1 ;NZ = '0' was pressed
00770 LD A,D ;get key pressed with '0', if so
00780 JR NZ,HTX
00790 JP 6167H ;back to Scripsit, if no '0' key
00800 ;
00810 HTX CP 'P' ;was it 'BP' ?
00820 JR Z,CLOKOD ;if so, enter CLOKOD
00830 CP 'T' ;same as above
00840 JR Z,CLOKOD
00850 CP '+'
00860 JR Z,CLOKOD
00870 CP ':'
00880 JR Z,CLOKOD
00890 CP '-'
00900 JR Z,CLOKOD
00910 CP 'x'
00920 JR Z,CLOKOD
00930 CP '.'
00940 JR Z,CLOKOD
```

```

F62D F62F 00750 CP '/'
F62F 284D 00960 JR Z,CLOMOD
F631 FE4F 00770 CP '0'
F633 285A 00980 JR Z,RSTIT ;restore tab-line
F635 FE49 00790 CP '1'
F637 2862 01000 JR Z,INIT ;initialize tab-line
F639 FE38 01010 CP 3BH ;else, if not 0...7 -> NOTHING
F63B 3008 01020 JR NC,NOTHC
F63D FE2F 01030 CP ZFH
F63F 3804 01040 JR C,NOTHC
F641 D630 01050 SUB 3BH ;0...7 pressed
F643 1803 01060 JR DSPUSK ;display that user key
F645 C34561 01070 NOTHC JP 6145H ;->Scripsit,if none of the
; 01080 ; above keys
F648 C5 01090 DSPUSK BC ;display user key routine
F649 D5 01100 PUSH DE ;save reg's
F64A E5 01110 PUSH HL
F64B 010000 01120 LD BC,0 ;initialize BC
F64E FE00 01130 CP 0 ;is it the termination char 0 ?
F650 280C 01140 JR Z,MULLX ;yes -> MULLX
F652 47 01150 LD B,A ;get 0 of user key to B
F653 210000 01160 LD HL,0 ;add 16 to HL 0 times
F656 111000 01170 LD DE,16 ;16 is the length of each user
F659 19 01180 ADRX ADD HL,DE ;key buffer
F65A 18FD 01190 DJNZ ADRX
F65C E5 01200 PUSH HL
F65D C1 01210 POP BC ;put result into BC
F65E 21AA6F 01220 MULLX LD HL,BUFR ;calculate start address
F661 09 01230 ADD HL,BC ;of user key buffer
F662 011000 01240 LD BC,16 ;search for 0 terminator
F665 3E00 01250 LD A,0 ;within this buffer
F667 ED81 01260 CPTR
F669 2B 01270 DEC HL
F66A 2B 01280 DEC HL ;points to last char in UK-buf
F66B 22AA6F 01290 LD (PTR),HL;-> PTR (key will be processed
F66E 3E01 01300 LD A,1 ;/backwards/ init. flag for
F670 32AA6F 01310 LD (FLAG),A;user key not yet processed
F673 3E00 01320 LD A,0 ;init. flag of
F675 3230F8 01330 LD (JMPFLG),A;count for backspaces
F678 E1 01340 POP HL
F679 D1 01350 POP DE ;restore reg's
F67A C1 01360 POP BC
F67B C36761 01370 JP 6167H ;back to Scripsit
; 01380 ; Processing of the actual user
; 01390 ; key will be taken care of by
; 01400 ; the intercepted screen-driver
; 01410 ;
F67E 08 01420 CLOMOD EX AF,AF' ;call to calc-mode: save A
F67F C031F8 01430 CALL CALCLP ;call it
F682 C37961 01440 JP 6179H ;back to Scripsit
; 01450 ;
F685 E5 01460 RSTIT PUSH HL ;restore tab-line
F686 D5 01470 PUSH DE ;it's a simple LDIR
F687 C5 01480 PUSH BC
F688 21AA6F 01490 LD HL,CURSAV
F68B 11803F 01500 LD DE,3FB0H
F68E 014000 01510 MOVEX LD BC,64
F691 ED80 01520 LDIR
F693 C1 01530 POP BC
F694 D1 01540 POP DE
F695 E1 01550 POP HL
F696 3E00 01560 LD A,0 ;simulate: no key pressed
F698 C36761 01570 JP 6167H ;back to Scripsit
; 01580 ;
F69B E5 01590 INIT PUSH HL ;save tab-line
F69C D5 01600 PUSH DE ;same as above
F69D C5 01610 PUSH BC
F69E 21803F 01620 LD HL,3FB0H
F6A1 11AA6F 01630 LD DE,CURSAV
F6A4 18EB 01640 JR MOVEX
; 01650 ;
; 01660 ;
F6A6 AA6F 01670 PTR DEFH BUFR ;pointer to actual byte pro-
; 01680 ; cessed by user key
F6A8 00 01690 FLAG DEFB 0 ;UK-processed flag
F6A9 00 01700 DEFB 0 ;safety UK buffer delimiter
F6AA 01710 BUFR EQU $
0080 01720 DEFS 128 ;USER KEY BUFFER
; 01730 ;
; 01740 ;

```

```

F72A 43 01750 ONLIN DEFH 'C' V: ACCU = 0 USER KEY
= 0 DEC. = 2
3A 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
F76A 54 01760 DEFH 'T' A0: 0.00 A1:
0.00
3A 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
41 30 3A 20 30 2E 30 30 20 20 20 20 20 20 20 20 20 20 20 20
20 20 41 31 3A 20 30 2E 30 30 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0040 01770 CURSAV DEFS 40H ;CURSOR LINE BUFFER
; 01780 ;
; 01790 ;
F7EA F5 01800 UKRD PUSH AF ;process user key call via 0x
F7EB 3AA6F6 01810 LD A,(FLAG)
F7EE FE00 01820 CP 0 ;is UK being invoked
F7F0 2837 01830 JR Z,OUT ;no -> OUT
F7F2 3A30F8 01840 LD A,(JMPFLG);backspaces processed
F7F5 FE00 01850 CP 0
F7F7 280F 01860 JR Z,CHRS ;yes, display next char
F7F9 3C 01870 INC A ;inc. backspace count
F7FA FE03 01880 CP 3 ;is it 3 yet ?
F7FC 2002 01890 JR NZ,KLAR ;no, backspace
F7FE 3E00 01900 LD A,0 ;yes, reset count to 0
F800 3230F8 01910 KLAR LD (JMPFLG),A;save count of backspaces
F803 F1 01920 POP AF ;perform
F804 3E9E 01930 LD A,9EH ;1 backspace
F806 B7 01940 OR A ;simulate: no key pressed
F807 C9 01950 RET ;back to Scripsit
; 01960 ;
F808 3C 01970 CHRS INC A ;display next char of UK-buffer
F809 3230F8 01980 LD (JMPFLG),A;inc backspace counter
F80C F1 01990 POP AF
F80D E5 02000 PUSH HL
F80E 2AA6F6 02010 LD HL,(PTR);get pointer to UK-char
F811 7E 02020 LD A,(HL) ;put into A
F812 FE20 02030 CP ' ' ;is it a space ?
F814 2808 02040 JR Z,OUT2
F816 FE00 02050 CP 0 ;or a 0 ?
F818 2807 02060 JR Z,OUT2 ;then terminate
F81A 2B 02070 DEC HL ;update
F81B 22AA6F 02080 LD (PTR),HL;pointer
F81E E1 02090 POP HL
F81F B7 02100 OR A ;clear flags
F820 C9 02110 RET ;return to Scripsit. Display
; 02120 ; char. in A
F821 3E00 02130 OUT2 LD A,0 ;reset
F823 32AA6F 02140 LD (FLAG),A;flag
F826 E1 02150 POP HL
F827 1801 02160 JR OK
; 02170 ;
F829 F1 02180 OUT POP AF
F82A 213640 02190 OK LD HL,4036H;back to Scripsit with no
F82D C36460 02200 JP 6064H ;key pressed
; 02210 ;
F830 00 02220 JMPFLG DEFB 0 ;counter for backspaces
; 02230 ;
; 02240 ;
; 02250 ;
F831 02260 CALCLP EQU $ ;calc-Subroutine
F831 F5 02270 SPZ PUSH AF
F832 C5 02280 PUSH BC ;save reg's
F833 D5 02290 PUSH DE
F834 E5 02300 PUSH HL
F835 D9 02310 EXX ;get cursor-
F836 22AA6F 02320 LD (CADR),HL;address of Scripsit
F839 D9 02330 EXX ; (in HL')
F83A C0FFB 02340 CALL UNPLUG ;disable disk BASIC exits
F83D C0ACFA 02350 CALL DSPBUF ;display CND-line
F840 C0AFA 02360 CALL CLRCHB ;clear CND-buffer
F843 21C23F 02370 LD HL,TAD ;clear Ti-field
F846 060F 02380 LD B,15
F848 C0A3FA 02390 CALL CLIT
F84B 2AA6F6 02400 LD HL,(CADR);point to char
F84E 23 02410 INC HL ;behind cursor position
F84F 11C23F 02420 LD DE,TAD ;and transfer num. value in
F852 C00AFA 02430 CALL TFER ;Scripsit text into Ti buffer
F855 C00DFA 02440 CALL TNDAR ;edit value with 4 dec. places
F858 00 02450 EX AF,AF'

```





F9D3 010010	04080	LD	BC,1000H	FAB5 CDC4FA	04890	CALL	DISPL	
F9D6 CD6000	04090	CALL	60H	FAB8 3E01	04900	LD	A,1	;STFLG=1 means 4 dec's
F9D9 C1	04100	POP	BC	FABA 18EC	04910	JR	ZMENT	
F9DA E1	04110	POP	HL		04920 ;			
F9DB 18EA	04120	JR	NTR	FABC 21C7FC	04930 TMR1	LD	HL,T01MS;transfer A0->A1 routine	
F9DD C35FFB	04130 ENTER	JP	GETKY ;back to input loop	FABF CDC4FA	04940	CALL	DISPL	
	04140 ;			FA92 21D43F	04950	LD	HL,AMAD	
F9E0 2185FC	04150 KSTTF	LD	HL,KTMS ;transfer V: to T: routine	FA95 11E63F	04960	LD	DE,ALAD	
F9E3 CDC4FA	04160	CALL	DISPL ;display CHD-line	FA98 010F00	04970	LD	BC,15	
F9E6 21C23F	04170	LD	HL,TAD ;clear T: field	FA9B ED80	04980	LDIR		
F9E9 860F	04180	LD	B,15	FA9D C35FFB	04990	JP	GETKY	
F9EB CDASFA	04190	CALL	CLIT		05000 ;			
F9EE 218E3F	04200	LD	HL,KAD ;transfer	FA00 21823F	05010 CLRCHB	LD	HL,CAD ;clear CHD-buffer	
F9F1 11C23F	04210	LD	DE,TAD	FA03 86AA	05020	LD	B,10	
F9F4 010A00	04220	LD	BC,10	FA05 3E20	05030 CLIT	LD	A,20H ;write (B) blanks into it	
F9F7 ED80	04230	LDIR		FA07 77	05040 CLCB	LD	(HL),A	
F9F9 C35FFB	04240	JP	GETKY ;back to input loop	FA0B 23	05050	INC	HL	
	04250 ;			FA09 10FC	05060	DJNZ	CLCB	
F9FC 2190FC	04260 ACOT	LD	HL,AMS ;transfer A0->T: routine	FA0B C9	05070	RET		
F9FF CDC4FA	04270	CALL	DISPL		05080 ;			
FA02 21D43F	04280	LD	HL,AMAD		05090 ;			
FA05 11C23F	04290 MOVE	LD	DE,TAD	FA0C 212AF7	05100 DSBUF	LD	HL,CHDLIN;display CHD-line routine	
FA08 010F00	04300	LD	BC,15	FA0F 11803F	05110	LD	DE,3F80H ;i.e. char's after C: in	
FA0B ED80	04310	LDIR		FA02 018000	05120	LD	BC,128 ;CHD-line	
FA0D C35FFB	04320	JP	GETKY	FA05 ED80	05130	LDIR		
	04330 ;			FA07 C9	05140	RET		
FA10 2190FC	04340 ACIT	LD	HL,AMS ;transfer A1->T:		05150 ;			
FA13 CDC4FA	04350	CALL	DISPL	FA08 21803F	05160 SBUF	LD	HL,3F80H;save CHD-line	
FA16 21E63F	04360	LD	HL,ALAD	FA0B 112AF7	05170	LD	DE,CHDLIN	
FA19 18EA	04370	JR	MOVE	FA0E 018000	05180	LD	BC,128	
	04380 ;			FA01 ED80	05190	LDIR		
FA1B 3E00	04390 BACK0	LD	A,0 ;back to Scripsit (cursor unchanged)	FA03 C9	05200	RET		
FA1D 3233FA	04400 BCOT	LD	(FEB),A		05210 ;			
FA20 C3D5FB	04410	JP	BACK ;back to Scripsit with:		05220 ;			
FA23 3E7B	04420 UPARR	LD	A,9BH ;cursor up 1 line	FA04 E5	05230 DISPL	PUSH	HL ;display message in (HL)	
FA25 18F6	04430	JR	BCOT	FA05 21823F	05240	LD	HL,CAD ;at C: field	
FA27 3E9C	04440 DOWNRR	LD	A,9CH ;cursor down 1 line	FA08 222040	05250	LD	(4020H),HL; on screen	
FA29 18F2	04450	JR	BCOT	FA0B E1	05260	POP	HL	
FA2B 3E1F	04460 TABI	LD	A,1FH ;cursor to next tab	FA0C CD6744	05270	CALL	4467H	
FA2D 18EE	04470	JR	BCOT	FA0F C9	05280	RET		
FA2F 3E3E	04480 BACKAR	LD	A,8EH ;cursor to start of line		05290 ;			
FA31 18EA	04490	JR	BCOT	FA00 860F	05300 TFER	LD	B,15 ;transfer 15 bytes	
	04500 ;			FA02 7E	05310 TF2	LD	A,(HL) ;from (HL) to	
FA33 00	04510 FCB	DEFB	0	FA03 FE20	05320	CP	' ' ;(DE) upto blank	
	04520 ;			FA05 2805	05330	JR	Z,OUTC	
FA34 21A6FC	04530 ROUND	LD	HL,RDMS ;round sel. accu to 2 dec's	FA07 12	05340	LD	(DE),A	
FA37 CDC4FA	04540	CALL	DISPL ;display CHD-line	FA08 23	05350	INC	HL	
FA3A CD1CFB	04550	CALL	TRANS ;sel. accu ->OP1; T1->OP2	FA09 13	05360	INC	DE	
FA3D 2171FB	04560	LD	HL,NULLST;simulate "OP1 + 0"	FA0A 10F6	05370	DJNZ	TF2	
FA40 118CFB	04570	LD	DE,TOKEN	FA0C C9	05380 OUTC	RET		
FA43 010300	04580	LD	BC,3		05390 ;			
FA46 ED80	04590	LDIR		FA0D 21C23F	05400 THOUR	LD	HL,TAD ;(HL)-> T: field	
FA4B 2178FB	04600	LD	HL,OP1 ;evaluate	FA00 CD650E	05410	CALL	0E65H ;doubl.prec. ASCII const->accu	
FA4E CD3723	04610	CALL	2337H	FA03 21C23F	05420	LD	HL,TAD ;clear T:	
FA51 2A7AFB	04620	LD	HL,(ACTACC)	FA06 860F	05430	LD	B,15	
FA54 E5	04630	PUSH	HL ;edit accu	FA0B CDASFA	05440	CALL	CLIT	
FA57 8609	04640	LD	B,9 ;with 2 dec's after ','	FA0B 21C23F	05450	LD	HL,TAD ;restore T: with	
FA5A 0E03	04650	LD	C,3	FA0E CD2FA	05460	CALL	ACDSP ;rounded value (4 dec's)	
FA56 3E80	04660	LD	A,80H	FA01 C9	05470	RET		
FA5B CD8E0F	04670	CALL	0FBEH		05480 ;			
FA5B D1	04680	POP	DE	FA02 E5	05490 ACOSP	PUSH	HL ;display ACCUM at place	
FA5C 010C00	04690	LD	BC,12	FA03 21A3FB	05500	LD	HL,SECACC; of sel. accu	
FA5F ED80	04700	LDIR		FA06 C0FF09	05510	CALL	09FFH	
FA61 3E20	04710	LD	A,' ' ;result back to sel. accu	FA09 8609	05520	LD	B,9 ;edit (ACCUM) numeric with	
FA63 12	04720	LD	(DE),A	FA0B 0E05	05530	LD	C,5 ;9 dec's , 4 dec's after ','	
FA64 13	04730	INC	DE	FA0D 3E80	05540	LD	A,80H	
FA65 12	04740	LD	(DE),A	FA0F CD8E0F	05550	CALL	0FBEH	
FA66 C35FFB	04750	CALL	UKTRF ;transfer to user key	FB02 D1	05560	POP	DE	
FA69 C35FFB	04760	JP	GETKY	FB03 010E00	05570	LD	BC,14 ;and transfer back	
	04770 ;			FB04 ED80	05580	LDIR	;from	
FA6C 21893F	04780 ZHEI	LD	HL,NKAD ;select 2 dec's user key routine	FB08 21A3FB	05590	LD	HL,SECACC; sel. accu to	
FA6F 77	04790	LD	(HL),A ;display '2' at 'DEC. ='	FB0B C0F709	05600	CALL	09F7H ; (DE) = T: field	
FA70 2181FC	04800	LD	HL,ZAMS ;display CHD-line	FB0E C9	05610	RET		
FA73 CDC4FA	04810	CALL	DISPL		05620 ;			
FA76 3E00	04820	LD	A,0 ;STFLG=0 means 2 dec's	FB0F 21A641	05630 UNPLUG	LD	HL,41A6H;disable DISK BASIC exits	
FA78 32A2FB	04830 ZMENT	LD	(STFLG),A	FB10 8615	05640	LD	B,15H	
FA7B C35FFB	04840	JP	GETKY	FB14 36C9	05650 LOOP	LD	(HL),0C9H	
	04850 ;			FB16 23	05660	INC	HL	
FA7E 21893F	04860 VIER	LD	HL,NKAD ;same as above with 4 dec's	FB17 23	05670	INC	HL	
FA81 77	04870	LD	(HL),A	FB18 23	05680	INC	HL	
FA82 2180FC	04880	LD	HL,VIMS	FB19 10F9	05690	DJNZ	LOOP	



```

FB1B C9      05700      RET
              05710 ;
FB1C 2A7FB  05720 TRANS LD      HL,(ACTACC); sel. accu -> OP1
FB1F 117FB  05730      LD      DE,OP1
FB22 C02FB  05740      CALL   MOVBR
FB25 21C2F  05750      LD      HL,TAD ; T: -> OP2
FB2B 11BDF  05760      LD      DE,OP2
FB2B C02FB  05770      CALL   MOVBR
FB2E C9      05780      RET
              05790 ;
FB2F 06F  05800 MOVBR LD      B,15
FB31 0E2D  05810      LD      C,'-' ;REPLACE - BY ITS TOKEN
FB33 7E    05820 TFLP  LD      A,(HL)
FB34 B9    05830      CP      C
FB35 2002  05840      JR      NZ,NTRLT
FB37 3EDE  05850      LD      A,00EH
FB39 12    05860 NTRLT LD      (DE),A
FB3A 23    05870      INC     HL
FB3B 13    05880      INC     DE
FB3C 10F5  05890      DJNZ    TFLP
FB3E C9    05900      RET
              05910 ;
              05920 ;
FB3F 3AA2FB 05930 UKTRF LD      A,(STFLG); transfer sel. accu to user k
FB42 FE00    05940      CP      0 ;is it 2 dec's ?
FB44 2008    05950      JR      NZ,KLTRF
FB46 0609    05960      LD      B,9 ;edit with 2 dec's after ','
FB48 0E03    05970      LD      C,3
FB4A 3E00    05980      LD      A,00H
FB4C C0BE0F 05990      CALL   0FBEH
FB4F 1803    06000      JR      KLSKP
FB51 2A7FB  06010 KLTRF LD      HL,(ACTACC); transfer sel. accu into
FB54 ED5B76FB 06020 KLSKP LD      DE,(USRKAD); USRKAD = buffer of selected
FB5B 060F    06030      LD      B,15 ; user key
FB5A 7E    06040 SKIP  LD      A,(HL) ; upto
FB5B FE20    06050      CP      ' ' ;blank
FB5D 2004    06060      JR      NZ,FDMUM
FB5F 23    06070      INC     HL
FB60 05    06080      DEC     B
FB61 1BF7    06090      JR      SKIP
FB63 12    06100 FDMUM LD      (DE),A
FB64 23    06110      INC     HL
FB65 13    06120      INC     DE
FB66 7E    06130      LD      A,(HL)
FB67 FE20    06140      CP      ' '
FB69 2802    06150      JR      Z,ENDTF
FB6B 10F6    06160      DJNZ    FDMUM
FB6D 3E00    06170 ENDTF LD      A,0
FB6F 12    06180      LD      (DE),A
FB70 C9    06190      RET
              06200 ;
              06210 ;
FB71 C0      06220 NULLST DEFB 00EH
FB72 30      06230      DEFB  '01'
              06240 ;
FB74 043F    06250 ACTACC DEFW 00AD
FB76 AAF6    06260 USRKAD DEFW 00FRT
000F          06270 OP1  DEFS  15
FB87 30      06280      DEFW  '00001' ;adjust for round off errors
              30 30 30 31
FB8C 00      06290 TOKEN DEFB 0
000F          06300 OP2  DEFS  15
FB9C 30      06310      DEFW  '00001' ; same as above
              30 30 30 31
FBA1 3A      06320      DEFB  ':'
FBA2 00      06330 STFLG DEFB 0
              06340 ;
              06350 ;
              06360 ;
FBA3 0000    06370 SECACC DEFW 0
FBA5 0000    06380      DEFW 0
FBA7 0000    06390      DEFW 0
FBA9 0000    06400      DEFW 0
FBAB 0000    06410 CADR  DEFW 0
0080          06420 BUFR DEFS 128
              06430 ;
              06440 CAD  EDU  3FB2H ;addresses of CMD-line fields
              06450 KAD  EDU  3FBEH
              06460 ACAD EDU  3F9FH
              06470 AMAD EDU  3FD4H
              06480 AIAD EDU  3FE6H

```

```

3FC2      06490 TAD  EDU  3FC2H
3FAE      06500 UKAD  EDU  3FAEH
3FB9      06510 NKAD  EDU  3FB9H
              06520 ;
              06530 ;
FC2D 73      06540 ADMS DEFW 'sel. Accu '
        65 6C 2E 20 41 63 63 75 20
FC37 03      06550      DEFB  3
FC3B 54      06560 TFMS DEFW 'T->sel.Acc'
        20 3E 73 65 6C 2E 41 63 63
FC42 03      06570      DEFB  3
FC43 61      06580 ADMS DEFW 'add '
        64 64 20 20 20 20 20 20
FC4D 03      06590      DEFB  3
FC4E 73      06600 SUBMS DEFW 'subtr '
        75 62 74 72 20 20 20 20
FCSB 03      06610      DEFB  3
FCS9 6D      06620 MUMS DEFW 'mult '
        75 6C 74 20 20 20 20 20
FC63 03      06630      DEFB  3
FC64 64      06640 DIMS DEFW 'div '
        69 76 20 20 20 20 20 20
FC6E 03      06650      DEFB  3
FC6F 73      06660 UKMS DEFW 'sel. UK '
        65 6C 2E 20 55 4B 20 20
FC79 03      06670      DEFB  3
FC7A 69      06680 IMMS DEFW 'input '
        6E 70 75 74 20 20 20 20
FC84 03      06690      DEFB  3
FC85 56      06700 KTMS DEFW 'V -> T '
        20 20 3E 20 54 20 20 20
FC8F 03      06710      DEFB  3
FC90 41      06720 AMMS DEFW 'A0 -> T '
        30 20 20 3E 20 54 20 20
FC9A 03      06730      DEFB  3
FC9B 41      06740 ALMS DEFW 'A1 -> T '
        31 20 20 3E 20 54 20 20
FCA5 03      06750      DEFB  3
FCA6 72      06760 ROMS DEFW 'round '
        6F 75 6E 64 20 20 20 20
FCB0 03      06770      DEFB  3
FCB1 55      06780 ZIMS DEFW 'Uk 2 Dec'
        68 20 32 20 44 65 63
FCB9 27      06790      DEFB  39 ;Apostrophe
FCBA 73      06800      DEFB  's'
FCBB 03      06810      DEFB  3
FCBC 55      06820 VIMS DEFW 'Uk 4 Dec'
        68 20 34 20 44 65 63
FCC4 27      06830      DEFB  39 ;Apostrophe
FCC5 73      06840      DEFB  's'
FCC6 03      06850      DEFB  3
FCC7 41      06860 T0IMS DEFW 'A0 -> A1 '
        30 20 20 3E 20 41 31 20
FCD1 03      06870      DEFB  3
5200          06880      END    START
00000 TOTAL ERRORS

```

```

AMAD 3FD4 AMMS FC90 AIAD 3FE6 ALMS FC9B ACAT F9FC
AC1S F913 AC1T FA10 ACAD 3F9F ACOSP FAF2 ADMS FC2D
ACTACC FB74 ADDI F93E ADMS FC43 ADRT F944 ADRCX F459
BACK FB05 BACK0 FA1B BACKAR FA2F BCOT FA1D BUFR FB0D
BUFR F6AA CAD 3FB2 CADR FBAB CALOUP FB31 CHRS FB08
CLCB F6A7 CLC00D F67E CLIT F6A5 CLNCMB F6A0 CHOLIN F72A
CURSAV F7AA DIMS FC64 DISPL FAC4 OIUP F977 BOHNR FA27
DSPBUF FAC DSPUSK F640 ENDTF FB6D ENTER F9D0 EVAL F94F
FDMUM FB63 FCB FA33 FLAG F6A0 GETKA FB62 GETKY FB5F
GDON F946 IDIT F69B DIMS FC7A INPUT FB83 JNFFLG FB30
KAD 3FBE KBDI F686 KBDINT F680 KLAR FB00 KLSKP FB54
KLTRF FB51 KSTTF F9E0 KTMS FC85 LOOP FB14 MOVBR FB2F
MOVE F6A5 MOVEX F68E MULT F96C MUMS FC59 NKAD 3FB9
NLT S208 NOTHC F645 NTRLT FB39 NULL F9A9 NULLST FB71
NULLX F65E OK FB2A OK1 F980 OP1 FB78 OP2 FB8D
OUT FB29 OUT2 FB21 OUTG FADC PTR F6A6 NDMS FCA6
REPT FB08 ROUND FAC4 RSTIT F685 SBRUF F6B8 SORMOD F680
SECACC FBA3 SKIP FB5A SLACC FB72 SP2 FB31 START S200
STFLG FBA2 SUBMS FC4E SUBTR F961 T0IMS FC77 T0M1 F68C
TABI FA2B TABTBL 7EB3 TAD 3FC2 TF2 FAD2 TFACCU F918
TFER FA08 TFLP FB33 TFMS FC38 TMDR F6D0 T0KEN FB8C
TRANS FB1C TX S219 UKAD 3FAE UMMS FCAF UPRED F683
UKRD F7EA UKSEL F982 UKTRF FB3F UNPLUC FB8F UPWR F623
USRKAD FB76 VIER FA7E VIMS FC8C WTR F9C7 MTX F611
ZHEI F6AC ZHEMT FA7B ZIMS FCB1

```

THE EXPLOITED MEDIA, AND HOW NOT TO BE ONE OF THEM  
A book review by exploited editor Jack Decker

Recently we were offered a review copy of THE UNABASHED SELF-PROMOTER'S GUIDE - What every man, woman, child, and organization in America needs to know about getting ahead by exploiting the Media by Dr. Jeffrey Lant. This is a very long title that describes the contents of a rather long book. Having waded about 2/3 of the way through its 366 pages, I know now why we got a review copy. We were just another media source to be exploited. Dr. Lant is hoping for just the sort of publicity I'm giving him by writing this column. And, I actually don't think he much cares whether I give the book a good or bad review (though I'm sure he'd prefer a good one), just as long as I give him some publicity. In the promotion business, getting your name known is half the battle.

So why am I letting Northern Bytes be exploited in this manner, especially when the book has (on the surface, at least) nothing at all to do with computers?

Well, first of all, many of our readers have secret dreams of writing a software package that will become as popular and well-known as Pac-Man or VisiCalc. Or perhaps you'd like to write a book on the rise and fall of the TRS-80 (with a chapter on NORTHERN BYTES included therein, I'm sure). You will need to promote it somehow and this book could be helpful in that regard.

Second, many of our readers have some connection with the media - either radio, television, or the press. If that describes you, you need this book in self-defense! There are thousands of self-promoters similar to Dr. Lant out there, just looking for ways to get free publicity. This book describes just about every "trick of the trade" of self-promotion that you'll ever come across. At least if you've read the book, you can recognize the various ploys when you see them coming.

And third, a self-promoter is a natural customer for word processing and computer services or equipment. Dr. Lant frequently advises readers to send out letters of various types, media releases, sample newspaper articles, and various other types of paperwork. In many cases the core of this material remains the same, but different versions are produced for different types of media, different localities, etc. The book gives numerous samples of various types of paperwork that the self-promoter must generate. Anyone that follows Dr. Lant's advice will either have to have word processing equipment (or hire someone that has it to do the work for him) or spend an awful lot of time retyping basically the same material over and over and over.....

Then, too, it is necessary for the self-promoter to keep records of various contacts he has made throughout the media. A natural function for a good relational database type program.

Okay, I have now successfully tied in the relationship of Dr. Lant's book to the readership of NORTHERN BYTES (which, if Dr. Lant were as good as he says he is, he would have made this connection for me in the materials he sent. But then, maybe he doesn't know that much about the capabilities of computers!). But, you may be asking, is the book worth the \$31.50 asking price? Well, I have to answer a qualified yes. The qualification depends on who you are, what you want to promote, and how hard you are willing to work at it. It also, to some extent, depends on your sense of what is right and wrong, what is acceptable behavior and what is not.

This last point deserves some attention. The term "unabashed self-promoter" is the key. Webster says that the term ABASH means "to make ashamed and uneasy; make self-conscious and embarrassed", so an UNABASHED self-promoter would be one who does not get embarrassed, ashamed, or self-conscious. And the word "self" is also important. The theme of this book is that you promote yourself first (as an expert in your field) and your book, product, service, etc. will be seen as more valuable because it has been produced by an obvious expert. Well, I have a couple of problems with this. Apparently Dr. Lant feels no shame or embarrassment in using his friends and associates to help promote himself, but I think most of us would draw the line at least in using our friends in this way. And then there is the whole matter of self-promotion, which more often than not turns into blowing one's own horn. At this point it might be appropriate to quote from the Bible: "Let another praise you, and not your own mouth; someone else, and not your own lips." (Proverbs 27:2, NIV). Or, as The Book puts it: "Don't praise yourself; let others do it!" The problem with praising yourself is that it leads to pride in one's self, and to once again quote from Proverbs, "Pride goes before destruction, a haughty spirit before a fall." (Proverbs 16:18, NIV).

Even if you feel that you can avoid the entrapment of pride (which isn't at all easy for most of us), there's still the matter of whether you have what it takes to be a self-promoter. If you follow

the advice in Dr. Lant's book, your name will be constantly appearing in newspapers and magazines and your voice and face will be familiar to radio and television audiences. Many of us wouldn't be comfortable with that kind of exposure. Some people thrive on being in the public view, however, and maybe you are one of them (or could learn to be).

Dr. Lant tells the reader a lot about himself in the pages of this book. Much of this information is designed to enhance his image as an "expert", but he lets us in on a lot of his personal views and feelings as well. This may have been a tactical error. About the only common ground that I could find with him is that he "feels uncomfortable in a suit or anything resembling standard business clothing." Other than that... well, I was always told that if you can't say something good about someone, don't say anything at all. The bottom line, however, is that Dr. Lant does not impress me as being a person I'd want to emulate or in any way pattern my life after. One particular quotation from page 93 of Dr. Lant's book stands out in my mind:

"The associates in your existing circle of friends, acquaintances and business colleagues belong to dozens of organizations. It is now your responsibility to find out which ones they are and to decide which of them are appropriate for you.

"I meet hundreds of people every year. With each of them I launch a networking conversation so that I can decide where they fit in my life and which contacts and associates they possess which may be of use to me. You do the same."

In other words, if I am reading correctly Dr. Lant is commanding me to choose my friends and associates according to what they can do for me, or how much use I may be able to make of them. No, thanks. I suspect that even if these people really wanted to be my friend to begin with, that desire would probably quickly leave them once I had "used" them a few times. Unless, of course, they figured that they could similarly use me to further their causes. Either way, it's not the sort of foundation I'd want to build a friendship on.

Also, it seems that at times Dr. Lant does not follow his own advice. At one point in the book, when writing about "Problem-Solving Process Articles", Dr. Lant advises the reader to "Keep the language simple, compelling. Readers rightly mistrust experts they cannot understand." Yet throughout the book (and especially in the first few chapters) Dr. Lant uses words that would probably not be in the vocabulary of someone that does not have at least a college education. Perhaps he is trying to impress us with his command of the English language, but it is my opinion that writers of self-help books should try to use language such that the average reader (who possibly has only a high-school education) will not have to sit with a dictionary at his side while reading the book.

So I have very mixed feeling about this book. I think it contains some very good advice and some simply awful advice, all within the same cover. And I do not think that it has the universal appeal that Dr. Lant seems to think it should have (I can't imagine too many children that would have the need or desire to exploit the media, for example). On the other hand, I would consider it "must reading" for editors (and other personnel) of magazines, newspapers, TV and radio news programs, etc. Between those two extremes, most of us would find the book interesting (and shocking in places!), but might have a tough time separating the "meat" from the "bones".

If, in spite of this review (or perhaps because of it?) you would like to obtain The Unabashed Self-Promoter's Guide, it is available for \$31.50 (price includes shipping) from Jeffrey Lant Associates, Inc., 50 Follen Street, Suite 507, Cambridge, Massachusetts 02138 or telephone (617) 547-6372.

MODEL III NEWDOS/80 ROUTINES

by Tony Domigan

P.O. Box 150, Thomastown, Victoria, Australia, 3074.  
MCI-ID : 254-5121

Here's some useful unpublished NEWDOS/80 version 2 (Model III) routines:

4C37H - Multiply H register by A register - result in HL  
4C39H - Multiply HL Register by A register - result in HL  
4C57H - Divide HL by 5, quotient in HL and remainder in A  
4C59H - Divide HL by A, quotient in HL and remainder in A  
44D2H - Convert DE to ASCII string pointed to by HL  
44D7H - Convert A to ASCII string pointed to by HL

## THE TRIANGLE NUMBER TRICK

[Dr. Michael Ecker, an Associate Professor of Mathematics and Computer Science at the University of Scranton, is a columnist/contributing editor of Creative Computing, Electronic Education and Soft Sector, and formerly of Byte, Popular Computing and Computer Gaming World. He writes columns on mathematical and computer recreations. A book based on material similar to that in his columns, "Gems of Recreational Computing", is scheduled for release by Arco Publishing in the Fall of 1985. He also does occasional freelance pieces on financial mathematics, and software reviews. As the alter ego of Recreational Mathematical Software, Dr. Ecker offers this serialization of some of his Magic Math Plus software, a collection of programs of "mathemagic", games, educational programs, financial utilities and numerous number tricks. At least one program from Magic Math will be offered in each installment. Readers who would prefer not to type in programs can purchase a disk directly from him. Northern Bytes is pleased to extend its coverage of useful information for TRS-80s to basic Basic applications and recreations. Information regarding questions for Dr. Ecker and/or ordering will be provided at the end of each article.]

Incidentally, for those of you who missed Dr. Ecker's column last issue, your editor must offer profuse apologies. We moved the deadline up one week at the last minute, without giving proper notice to Dr. Ecker. This installment of Magic Math Plus arrived in plenty of time for the old deadline. It was your editor's fault only, and I will accept being stoned with abacus beads (preferably plastic ones!) as my punishment.]

The Triangle Number Trick is a cute program named for the configuration achieved by the numbers used. Readers may find it reminiscent of Pascal's triangle. Indeed, the trick is based on a theorem which is both combinatorial and number-theoretic in nature. Roughly, it says that binomial coefficients  $C(p,k)$  - the number of ways of choosing  $k$  things from  $p$  without regard to order - are all divisible by  $p$  if  $p$  is prime. This program exploits the  $p=5$  case.

It doesn't really matter whether or not any of the above makes any sense to you. This is because the program is self-contained. Furthermore, if you are unfamiliar with Pascal's triangle, one of the programs on the software I am marketing contains a tutorial on it. In fact, readers who desire a fuller explanation of the Triangle Number Trick may purchase Magic Math Plus, which contains this trick's program with full explanation. The software containing the Triangle Number Trick is also accompanied by numerous other mathematical recreations and bonuses. Furthermore, Northern Bytes readers may obtain Magic Math Plus for 20% off. See the note at the end of this article for more information.

Triangle will run on any level 2 system (Model 1,3,4, tape or disk), possibly others. The program begins with your TRS-80 predicting, before you do anything, what will be the final answer to the calculations which are going to be performed. What is remarkable about this is the fact that the calculations are based on your inputs to a very large degree. Nevertheless, in spite of this, the computer provides you with the correct answer in advance. The computer selects two integers (whole numbers) and then requests that you provide four integers of your own choosing. The computer then displays the six numbers on a line.

Next, your 80 will add every two adjacent pairs of numbers and, via some more liberal use of the Print @ command, print the sum under and in between the two numbers. If the sum is larger than 5, the computer will only display the result for a brief moment, and then will subtract 5. It will keep doing this until the result is no longer greater than 5. So, if the two numbers are 13 and 8, for example, the computer will display 21 (the sum) under and between the 13 and the 8, but only for an instant, as 21 is larger than 5. Then, 21 will vanish and be replaced by 16 (as  $21-5=16$ ). Since 16 is still larger than 5, it too will vanish and in succession you will see 11, then 6, and finally 1.

This process is repeated for every pair of adjacent numbers in the top row until a second row of answers appears. This row will then contain five numbers. Now what? Well, the computer now repeats the process using pairs of adjacent numbers from the second row. This in turn yields a third row of four numbers, to be followed by a fourth row of three numbers, a fifth row of two numbers, and then - the moment of anticipation - the final row bearing the final answer... the one somehow picked by the computer!

I have little doubt that many readers will have little or no trouble figuring out what the computer does, but this still does not fully explain why the trick works. Ah, but you will allow me that one small mystery - at least for a little while - won't you? For now, enjoy the trick! See if you can't figure it out for yourself!

I actively solicit your comments, pertinent questions, suggestions, improvements (always plenty of room for them!) and so on. Write to me directly at the address above. For those who own TRS-80 Model 3, 4 or 4P and who would prefer not to type in the program, send me a disk and \$7.50, or \$10 alone, and I'll include the program, plus an extra half dozen programs in a menu-generated format on a self-booting disk with a licensed DOS. Only one disk drive is needed. Ask for Magic Math Plus sampler disk with Triangle Trick. If you don't ask for Triangle Trick explicitly, it will not automatically be included as it's not ordinarily on the demo disk, so please do ask! You may also obtain Triangle Trick, previous issue's Super-Blackjack, and earlier issue's loan amortization, as part of the 36 program, double volume collection available on Magic Math Plus.

Magic Math comes on a self-booting disk with XDOS, a licensed disk operating system compatible with TRSDOS 1.3, from Mr. Jim Kyle of the Software Factory. The collection is totally menu generated (again, courtesy of software from the Software Factory - and available from Recreational Mathematical Software). This combination makes use of Magic Math almost automatic.

Volume 1 contains approximately 20 programs, including Fastloan, Compound Interest, Super-blackjack, Super-Trick, Super-Fast Prime Number Cruncher, The Game of N, Fibonacci Numbers, Additive Sequences, Base Two (a trick with explanation) and loads of other goodies!

Volume 1 sells for \$24.95 plus \$1 for shipping and handling. A volume 2, written by programmer/writer Mr. David B. Lewis (who writes in the likes of 80 Micro and others), is available with lots of other numerical curiosities involving convergence, as in the Collatz conjecture, and with graphic and other games (as in Hexapawn, Repeater, etc.), with at least 15 programs available for the same price. Same menu generated format available for Volume 2.

Special offer! Volumes 1 and 2 combined, with the self-booting disk, XDOS, a minimum of 36 programs (including some bonuses and tutorials) in a menu generated format, a title page and four menu programs / pages - usually all for \$36 + \$1 S&H - now 20% off! Send just \$29.75 - postpaid. That's just \$4.80 above the cost of either volume alone.

Lastly! I will be pleased to send anybody who wishes further information a catalog if you send a self-addressed stamped envelope. Please specify your computer brand/model. I remind you again that your comments and questions are welcome and appreciated. Until next issue! Happy Computing!

Dr. Michael W. Ecker  
Recreational Mathematical Software  
129 Carol Drive  
Clarks Summit, Pennsylvania 18411  
(717) 586-2784

[Screen dump of program, just before final answer (3 in this case) is placed at bottom point of triangle. In this example, the computer provided the numbers 8 and 35, while the user provided the numbers 41, 27, 15, and 38. The computer predicted that the answer would be 3 before the user's numbers were entered!]

```

      8      41      27      15      38      35
    4      3      2      3      3
      2      5      5      1
          2      5      1
              2      1

```

THE CORRECT ANSWER WILL BE 3

THE COMPUTER WILL NOW ADD ADJACENT NUMBERS AND, IF NECESSARY, SUBTRACT 5 UNTIL 5 OR LESS IS OBTAINED. IT WILL THEN PLACE THE RESULT UNDER AND BETWEEN THOSE TWO NUMBERS, FOR EACH PAIR.

[Program listing:]  
100 CLEAR500:S\$=STRING\$(23,42)  
110 CLS:PRINT@342,S\$:PRINT@406,"\*TRIANGLE NUMBER TRICK\*":  
PRINT@470,S\$:  
115 PRINT @960,"RECREATIONAL MATHEMATICAL SOFTWARE (C  
) 1985 DR. MICHAEL ECKER"

```

120 FORJ=1TO1500:NEXTJ
200 CLS
205 PRINT:PRINT
210 PRINT "In this trick, the computer will display numbers you INPU
T as"
220 PRINT "the top row of a triangle. It will add adjacent numbers a
nd"
230 PRINT "then subtract multiples of 5 until the answer is 5 or less
."
240 PRINT "Each answer will be placed between and below the numbe
rs so"
250 PRINT "as to form a new, smaller row of numbers."
260 PRINT "This process will be repeated until a single number is ob
tained."
270 PRINT "Before you even begin INPUTing the numbers, the compu
ter"
280 PRINT "will provide what will ultimately be the final answer!"
290 PRINT
300 INPUT "PRESS <ENTER> WHEN YOU ARE READY TO CONTINUE
"IX
305 DIM A(100)
310 RANDOM
320 A(1)=RND(40):A(6)=RND(50)
330 Q=A(1)+A(6):Q=Q-5*INT((Q-1)/5)
340 CLS
350 PRINT (512,CHR$(23))"THE CORRECT ANSWER WILL BE"Q:FOR
Z=1 TO 2000:NEXTZ
355 PRINT (630,CHR$(28))"
360 PRINT CHR$(28)"THE COMPUTER HAS NOW PICKED TWO NUMB
ERS"FOR Z=1 TO 1500:NEXTZ
365 PRINT CHR$(28)"
366 PRINT (5,A(1)):PRINT (55,A(6)):FOR Z=1 TO 1200:NEXT
370 PRINT (640,"PLEASE INPUT FOUR WHOLE NUMBERS (SAY, 5 TO
50) ONE AT A TIME"
380 PRINT (704,"HIT <ENTER> AFTER EACH NUMBER"
390 INPUT A(2):PRINT (15,A(2)):PRINT(767,"":INPUT A(3):PRINT (25
,A(3)):PRINT (767,"
395 INPUT A(4):PRINT (35,A(4)):PRINT (767,"":INPUT A(5):PRINT (4
5,A(5))
400 PRINT (768,"
410 PRINT (640,"THE COMPUTER WILL NOW ADD ADJACENT NUMB
ERS AND, IF NECESSARY, "
420 PRINT "SUBTRACT 5 UNTIL 5 OR LESS IS OBTAINED, IT WILL
THEN PLACE"
430 PRINT "THE RESULT UNDER AND BETWEEN THOSE TWO NUMB
ERS, FOR EACH PAIR."
440 PRINT
450 INPUT "PRESS <ENTER> WHEN YOU ARE DONE READING THIS"
IX
460 PRINT (932,"
470 FOR I=7 TO 11
480 A(I)=A(I-6)+A(I-5)
490 READX
510 PRINT (X,A(I))
520 IF A(I)>5 THEN GOSUB 800:GOTO 510
530 NEXT I
540 FOR I=12 TO 15
550 A(I)=A(I-5)+A(I-4)
560 READ X
570 PRINT (X,A(I))
580 IF A(I)>5 THEN GOSUB 800:GOTO 570
590 NEXT I
600 FOR I=16 TO 18
610 A(I)=A(I-4)+A(I-3)
620 READ X
630 PRINT (X,A(I))
640 IF A(I)>5 THEN GOSUB 800:GOTO 630
650 NEXT I
660 FOR I=19 TO 20
670 A(I)=A(I-3)+A(I-2)
680 READ X
690 PRINT (X,A(I))
700 IF A(I)>5 THEN GOSUB 800:GOTO 690
710 NEXT I
800 IF A(I)>5 THEN A(I)=A(I)-5:FOR Z=1 TO 500:NEXT:RETURN
980 FOR Z=1 TO 1200:NEXT
990 PRINT (640,"OKAY, AS PREDICTED, HERE COMES THE ANSWER
OF"Q:"
1000 PRINT (704,"
1010 PRINT (768,"
1020 PRINT (932,"

```

```

1025 FOR Z=1 TO 1500:NEXT
1030 Q=A(19)+A(20)
1040 PRINT (350,Q)
1050 IF Q>5 THEN Q=Q-5:FOR Z=1 TO 500:NEXT:GOTO 1040
1060 FOR Z=1 TO 1500:NEXT
1070 PRINT (948, "TO PLAY AGAIN, TYPE THE LETTER P"
1100 IS=INKEY$:IF IS="" THEN 1100 ELSE IF IS="P" THEN RUN
1200 DATA 74,84,94,104,114,143,153,163,173,212,222,232,281,291

```

## ATTENTION HARDWARE HACKERS

As this issue of Northern Bytes was going to layout, I got a letter from Don McKenzie, 29 Ellesmere Crescent, Tullamarine, Victoria 3043, AUSTRALIA. Don had read my plea for a low-cost 1200 baud MODEM and wrote to advise me that Mick Gulovsen sells such a unit in kit form for \$130 Australian (The Australian dollar is worth around 62 cents at present, so at the time of this writing such a modem could be purchased for somewhere around \$80 U.S. plus shipping).

The kit uses an Am7910 WORLD-CHIP<sup>tm</sup> Modem IC, op amp, and 2716 for control logic switching. The Am7910 is capable of operation using both the Bell and CCITT standards, so the modem should be usable within the U.S. (possibly with a bit of very minor modification - I haven't seen the kit so can't say whether any modification would be necessary or not). It does not include front panel switches, LED, or phone jacks, and does not have an auto-answer feature. It is designed to mount inside your computer (or expansion interface) and draw power from your computer's power supply, but an external power source (using a 7905 regulator) may be used if necessary.

Of course, such a unit would not be F.C.C. registered and thus could not be legally connected to the telephone system, but if you have your own PBX system (or like to live dangerously and figure it's none of the phone company's business what you connect to your phone lines), you may want to contact Mick Gulovsen, 14 Sutherland Street, Glenroy, Victoria 3046, AUSTRALIA for more information.

Don has also been busy lately. He has developed a 8K-4 Centronics parallel to serial converter. It connects to a Centronics line to your printer using standard Centronics male and female connectors. It uses a 280A CPU and can be configured for 8K, 16K, 32K, or 64K simply by adding extra 4164 (64K) memory chips. Don is selling what he calls a "PBUFF SHORT FORM KIT" which consists of a bare single sided printed circuit board and one 2716 EPROM programmed with PBUFF software (version 1.0). You also get full assembly instructions (including debugging information in case you get into trouble) and free hardware debugging advice via phone or mail. You must provide the other parts - a suitable case, power pack (+5 volts stolen from your computer or a cheapie external power supply capable of putting out 9 volts AC or DC at 400 ma or more), Centronics male and female connectors, as many 4164 memory chips as you care to use, and about \$20 worth of additional components.

The PC board is designed to mount directly into a DICK SMITH ELECTRONICS catalog # H-2505 instrument case. Dick Smith Electronics has stores in Australia and New Zealand and has just recently started opening stores and "authorized re-sellers" here in the U.S. (they had a number of pages in a recent issue of Radio-Electronics magazine).

Don sells the PBUFF kit for \$35.00 Australian (around \$22 U.S.) plus \$5 for postage to North America. If you need a 3.58 MHz crystal, Don can supply that for \$2.00. He also has a supply of 4164 memory chips available but as he notes, the prices on those fluctuate on a daily basis (and most likely are sold for less here in the U.S.).

Don also offers several other hardware modifications for the TRS-80 Model I and the "clones" (PMC-80, Dick Smith SYSTEM-80, Video Genie, etc.). He puts out a catalog but as he is basically a hobbyist (not a large corporation) and airmail postage is expensive, I'd suggest you include two or three dollars for postage if you want the catalog. If you tell him what kind of system you have (or what you might be looking for in the way of hardware mods) he may be better able to advise you as to what he has that might be interest to you. Once again, Don's address is: Don McKenzie, Ellesmere Crescent, Tullamarine, Victoria 3043, AUSTRALIA and his telephone number is (03) 338-6286 (from the U.S. dial 011+61+3+338-6286, and remember the time difference - no one likes to be awakened at 3 A.M.)

## THE FUTURE OF COMPUTER TELECOMMUNICATIONS by Jack Decker

When you stop and think about it, many "advances" in science and technology are very simple once you understand how they work. For example, the basic idea behind the telegraph is so simple that just about any junior high school science student can put a crude model together using blocks of wood, nails, wire, strips of metal cut from tin cans, and a battery. Yet we still remember the name of Samuel Morse as the inventor of the telegraph.

Or, take a more recent example. Today's communications satellites, which orbit in a fixed position above the equator, are said to be located in the Clarke Orbit Belt. This belt is named after Arthur C. Clarke, a science fiction writer who first envisioned the belt of satellites orbiting above the earth in a geosynchronous orbit (that is, each satellite appears to be in a stationary position in the sky, as viewed from the earth). The idea of a band of orbiting satellites is one that any high school science student can understand, yet it was significant enough to have Mr. Clarke's name forever attached to the satellite belt above the equator.

Point is, some ideas seem so obvious once you think about them a little, yet they revolutionize the way we live once they are implemented.

Well, I have a modest proposal. This idea is also one which seems simple once you think about it, yet it could revolutionize communications. Moreover, it uses no new technology. The system could be implemented on existing hardware, once the software were written.

Please note that the benefits of this system are partly artificial. If the pricing structure of telecommunications services (particularly the long distance telephone network) were different than at present, there might be no need for this proposal. The one thing that could eliminate the usefulness of this proposal is if telephone charges were not distance-sensitive (in other words, if the charge were the same to call a telephone 20 miles away or 3000 miles away). Although some feeble motions in that direction have been made by some of the long distance carriers, I really don't foresee the elimination of distance-sensitive pricing in the near future. In addition, new technology (a two-way cable or satellite link to every home, for example) could eliminate the need for what I propose. But then, Morse's telegraph had a limited lifespan, yet it was still the only practical means of long distance communications for a number of years.

So, with that in mind, here's the proposal:

At the present time, most computer services (those provided via mainframe computers) are accessed via the telephone network. That is, you sit at your computer, and you (or your "smart" telephone modem) dial a number that connects you with the computer. Now, if the computer in question happens to be in the same city that you're in, there are no long distance phone charges incurred by either you or the host computer. This is the ideal situation (remember this, we'll be coming back to it).

However, let's suppose you're in Chicago and the computer you want to access is in New York. At this point you will probably have two choices. One choice, which is always available, is to simply dial up the computer over a regular long distance network, such as AT&T, MCI, GTE Sprint, or any of the other numerous long distance carriers. In this case your call will be charged according to both time and distance - in other words, you will pay for each minute online, and the charge per minute will be based on the distance between you and the mainframe computer. However, you will NOT be charged according to the number of characters transmitted and/or received, nor will you incur a surcharge for using a higher baud rate.

Your second option, assuming that you are communicating with a major supplier of computer services (an information utility, for example) is to communicate using a packet network. In this case you would dial a local number in Chicago, and be connected to a small "front-end" computer which would assemble your messages into data "packets" for transmission at very high speed to the "host" computer. I'm not going to get into a discussion of packet network theory here, but the bottom line is that from the Chicago user's standpoint, he has made a local telephone call to access the computer in New York. However, the company which owns the computer in New York is billed by the packet network according to time and the number of characters transmitted and/or received. The computer services company in New York usually turns around and bills this back to our Chicago user on his monthly statement. HOWEVER, because the formula for calculating characters transmitted/received can be a bit complicated, most computer services companies (especially the information utilities) simply opt

to bill their customers for packet network usage on a per-hour basis. The per-hour rate is then based on the packet network's per-hour charge, the average user's per-hour character usage multiplied by the packet network's character charge, plus a little profit for the computer services company.

Let's review for a moment. If the computer to be accessed is local, there is no charge for time, distance, or characters transmitted/received (except perhaps in some major metropolitan areas where "measured service" is in use). If the computer is accessed via the long-distance telephone network, then we are charged for time and distance but not for characters transmitted/received. If the computer is accessed via a packet network, then we are charged (directly or indirectly) for time and characters transmitted/received but not for distance.

But even in the case of packet networks, distance can be a factor. Certainly the packet network must pay for the lines it leases from other communications carriers according to distance, and that affects the rates they must charge. Also, it may be a toll call to the nearest packet network node, and there will be an additional charge if access is made from a very distant point (another country, for example).

One thing that does NOT change whether long distance telephone or packet switching networks are used is the amount of information transmitted from New York to Chicago. If the amount of information to be sent could be reduced, the charges would naturally be less.

One suggestion that immediately comes to mind is text compression - that is, compress all common words (of the English language, for example), to two-byte codes and send only the codes. This is worth considering but is difficult to do in practice, especially on a public system where various users may wish to send vastly different types of information.

However, larger savings are possible on most public-access systems. In order to understand this, let's visualize a system that many home computer users are familiar with - the Compuserve Information Service.

On Compuserve, almost all of the information is designated by "page numbers". For example, at a given prompt a user might say GO PCS-1 which would display the information on that page. Now, the information on that page might change on an hourly basis. Then again, it might not be changed for several months. Let's suppose a user in New York connects to Compuserve (which is located in Ohio) and reads that page. Whether he is making a regular long-distance phone call or using a packet network, the information on that page is transmitted, character by character, from Ohio to New York.

Now suppose our New York user calls back the next evening and for some reason reads the same page, which, as it happens, has not been changed since the last time he read it. The page is again transmitted from Ohio to New York, character by character, and again the long distance company or packet network makes money on it (Compuserve happens to own its own packet network, which is unusual, but that really doesn't affect the principle being discussed here). Now suppose 50 other New Yorkers call Compuserve that same evening and happen to read the same page. Again it is transmitted between Ohio and New York fifty more times. Somebody makes money on all that repetitive transmission!

"But", one might wonder, "computers are getting cheaper. Memory is getting cheaper. Wouldn't it be more cost-effective to put a computer in New York, let it store each of the 'pages' and transmit them to local users as required, and only transmit a 'page' from Ohio to New York when the page is updated?"

Well, why not? The more information that can be provided locally, without accessing the host computer in Ohio, the closer we can get to our "ideal situation" of not having to pay for time or distance.

Perhaps what we are talking about here is the next generation of "front-end" computers. In present terminology, the "front end" computer is basically a "concentrator" for a larger host computer. For example, there is, no doubt, a "front end" computer in, say, Chicago (and in every other major city), that has maybe 24 (to pick an arbitrary number) modems connected to it. In fact, each of the packet networks will have a "front end" computer of this type (it's sometimes called a "pad"). As users dial into the modems, the "front end" computers handle the data communications to and from each individual modem. The "front end" then "concentrates" all this data and transmits it (at a much higher baud rate) over the packet network's leased circuits. Today's "front end" computer, however, does little or nothing with the actual data. It simply re-transmits it at a higher or lower baud rate (well, yes, it's actually a little more intelligent than that, and can usually be configured to adapt itself to the end user's terminal requirements, but that's about it).

My idea is to give the "front end" computers a lot more to do. In the system I envision, each "front end" computer would provide as much of the computing power as possible, without accessing the "host" mainframe computer except when absolutely necessary.

Before I go on, however, a word about terminology. A few paragraphs back I talked about using "pages" of information. I used this example because the concept is familiar to many computer users, especially those that access Compuserve. With other services, other ways might be used to identify a given block of text. Text or data can be transmitted using "pages", "messages", "packets", or even "blocks of memory" in the main "host" computer as the logical divisions. The basic idea is that once a given block of text or data has been sent to a "front end" computer, it is stored there for future access and not re-transmitted by the host computer unless it has been changed or modified.

Let's jump a step further. Suppose we have a main "host" computer, located in New York, and, say, 500 "smart front-end" computers located in major cities around the country. Let's further suppose that our "host" computer is running an electronic mail and bulletin-board type system. In this case, our blocks of text - the "coin of the realm", as it were - will be messages. Once a message is transmitted to a given "front end" computer, it will not be re-transmitted unless the message is changed. How will this save transmission time? Two ways immediately come to mind:

1) In the case of public (bulletin board) messages, once a message has been transmitted to a given local "front end", it would be stored there for reading by other local area users. It could be read hundreds of times, but would only have been transmitted over the long distance or packet network once.

2) In the case of a private message, it would be available for re-reading as many times as necessary, even though it was transmitted to the "front end" only once.

I've deliberately avoided going into specifics about the system because the implementation would necessarily be varied depending on the type of service and software available (though using "packets" would be an option no matter what the end use). But to give an example, let's suppose an electronic mail user wants to read his mail. Let's further suppose that he has four messages waiting, two of which have been previously transmitted to the "front end" (either on the request of that user or perhaps another local user who got a "copy" of those two messages). The dialogue might go like this:

1) User logs in. Login information transmitted to host computer for verification.

2) Host OK's login, and searches message base. It discovers that the user has four messages waiting. The message numbers are 10234, 11442, 11616, and 12092. Host transmits this information to "front end".

3) "Front end" computer searches its memory bank and discovers it already has messages 10234 and 11616. It sends request to host to transmit messages 11442 and 12092, which it stores for possible future use.

4) "Front end" transmits all four messages to electronic mail user, plus it handles all the normal menu options and other "busy work", inquiring of the host only when it needs data it does not already have or data that may have changed.

5) When user logs off, "front end" computer transmits to host the associated billing data plus the information that the user has now read the four messages (if in fact he did read them).

6) When messages are deleted from the system, the host computer transmits this information to the "front end" computers as a background task (done during "non-busy" intervals).

At this point you may be wondering what happens when the memory in a "front end" computer is full (assuming that the "front ends" have less memory capacity than the host). Simple, messages are dropped. Either the oldest messages or the least-frequently-read messages can be dropped. If someone requests one of these dropped messages, it can always be retrieved again from the host.

Note that there are a lot of possible permutations of this system. For example, a system could be designed where the "front end" computer only connects with the host for, say, five minutes out of each hour. This type of setup might be very cost-effective for a service that does not require instantaneous delivery. For example, on a bulletin board type service, the fact that others might not be able to read a message until an hour or two after it is posted might be quite acceptable, if the cost of the service can be lowered. In this case each "front end" computer could store the entire current message base, receiving an update from the host (and transmitting new messages it has received to the host) once an hour. For a hobbyist type system, this could even be stretched out to once a day, with communications between the "front ends" and the host taking place only during off-peak rate periods.

When you consider that the same block of text may be transmitted from place to place on a repetitive basis, it makes sense that if these repeat transmissions can be avoided, substantial savings in costs are possible. If you use an information utility such as Compuserve, think how many times some of the menus must be re-transmitted. If you live in a large city, these same menus may be re-transmitted to your area many times a day. This may make the phone companies and packet network people happy, but not the end user when he gets the bill. So, why not transmit the information once, then store it in a computer in the local area for additional accesses?

That's about as far as I intend to take this idea right now. Actually, I can't believe that I'm the first person to propose something like this. The idea seems so simple, so obvious. But, just in case... if anyone gets the notion to name this system after me, keep in mind that I never went to college and wouldn't be caught dead (or alive) in a necktie. Is that really the type of person whose name you want mentioned in tomorrow's computer textbooks?!

#### ZAPS

Provided and translated by Paul Fransen

The program THE ARRANGER is a very nice program to catalog your programs on diskette. The only problem is that the program assumes that on a Model 3 you use disk drives with a track stepping rate of 3 ms. If you have older (and slower) drives, then there could be trouble. However, you can fix it with the following zap:

Track 3, sector 8, byte 77: E6 FC becomes F6 03.  
(provided by Karel Honings, Holland).

If you have the programs CODIR/CMD and DIS/CMD or DIC/CMD then you may find that the following zap can be handy!

CODIR/CMD, sector 11, byte 8A: 52 becomes 53  
byte ED: 52 becomes 53

Now CODIR works with DIS/CMD instead of directly with DIR, so the files will be sorted before they appear on the screen (provided by Stef Taartmans, Holland).

You can do the same thing with CAT/CMD and DIS/CMD!

CAT/CMD, sector 3, byte FC: 52 becomes 53

You scroll in Superzap with the + and - keys. It is nicer to do this with the arrow keys!

SUPERZAP/CMD, sector 3, byte BD: 3B becomes 5B  
sector 4, byte 17: 2D becomes 0A  
byte 35: 2D becomes 0A

(provided by Joop van Dam, Holland).

## 64K

## \$59.95 PPD

### INSTALLED IN KEYBOARD

#### TRS-80® Model I-LII

Send us your Keyboard and we will convert it to full 64K memory (48K RAM). Improved performance with or without Interface. 90 day warranty. Satisfaction guaranteed. Quick return. Free return freight within U.S.A.

#### ICE

International Carbide & Engineering, Inc.  
100 Mill St. • P.O. Box 216  
Drakes Branch, VA 23937

(804) 568-3311 TWX: (910) 997-8341

®TM TANDY CORP.



**SCRPRT FAST SCREEN PRINT PROGRAM**  
for Radio Shack dot matrix printers  
by Serge Y. Calmettes

Have you ever worked on a fancy screen graphics display, and wished that you could make a printed copy of it? Well keep on reading because SCRPRT does exactly that!

The program works with the TRS-80 Models III and 4 (in the Model III mode) [Model I users, see editor's note at end of this article], with or without disk and with the following line printers: LP V, VII, VIII, and DMP 100, 200, and 400.

SCRPRT can be activated either from the disk operating system (DOS), or from BASIC by pressing simultaneously the up and down arrow keys.

A full screen is printed in 30 seconds or less. This program is transparent to the user and can be activated at any time from DOS or BASIC.

Once the program is installed, the keyboard driver is intercepted, and therefore SCRPRT can be triggered anytime that the keyboard is interrogated. SCRPRT scans the video memory, and the printer mode is switched from characters to graphics on a line per line basis as required by the type of display. When the last line is printed, the operation returns to the program that was running when SCRPRT was called in.

Use EDTASM or an equivalent program to assemble SCRPRT (listing 1). You can also use a monitor such as DEBUG, MON3, or TASMOM to enter the machine code. For 48K systems the start address is 0FE80H; the end address, 0FFF6H; and the entry point, 0FE80H. At any rate save SCRPRT on disk or cassette. From DOS, to install SCRPRT, type SCRPRT<enter>; from BASIC load and run SCRPRT using the SYSTEM command.

Listing 2 lets you enter the program from Level II BASIC. Make sure to SAVE the program before running it - a typing error can have some very unpredictable effects with machine language programs! The disk version is self-protecting, but not the Level II BASIC one (answer 32382 to the MEMORY SIZE prompt if under Level II BASIC).

At times, a left bracket is displayed and printed when the up and down arrow keys are depressed. If you find this objectionable, instead use the down arrow and space bar keys. Change line 470 of listing 1 to:

```
CP 90H ;DOWN ARROW & SPACE BAR
or line 150 of listing 2 to:
```

```
DATA 174,025,058,064,056,254,144,194,073,000,245,197
```

Now you will be able to enjoy all those fancy displays even when your computer is turned off!

[EDITOR'S NOTE: Model I users should be able to make this program run under disk-based systems (not Level II) by changing the following four lines of listing 1:

```
00200 VOLINE EQU 4467H ;WORKS UNDER DOS ONLY!!
00210 CKPRT EQU 05D1H
00240 MEMPRT EQU 4049H ;xxx FOR DISK ONLY xxx
01880 LD (37EBH),A ;PRINT IT
```

Model I cassette-based system users can also use the above changes except that line 200 should be changed as follows:

```
00200 VOLINE EQU 2875H ;VALID UNDER BASIC ONLY
```

Please note that the above changes have not been tested, but should be all that is required for Model I users!]

#### LISTING 1

```
00100 ;SCRPRT
00110 ;
00120 ; BY SERGE Y. CALMETTES, MAY 1982
00130 ;FOR TRS-80 MODEL III
00140 ;TO PRINT CONTENTS OF SCREEN, INCLUDING
00150 ;GRAPHIC CHARACTERS
00160 ;xxx SEE REMARKS FOR LEVEL II NON-DISK SYSTEMS xxx
00170 ;
FE80 00180 ORG 0FE80H ;0FE80H FOR 32K
0049 00190 KEYSON EQU 0049H
0218 00200 VOLINE EQU 0218H
0448 00210 CKPRT EQU 0448H
3840 00220 KEYS EQU 3840H
4016 00230 KYPNTR EQU 4016H
4411 00240 MEMPRT EQU 4411H ;xxx FOR DISK ONLY xxx
00250 ;
FE80 2A1640 00260 SETUP LD HL,(KYPNTR) ;SET POINTERS
```

```
FE83 2289FE 00270 LD (KYDNR+1),HL
FE86 2185FF 00280 LD HL,PRINT
FE89 2212FF 00290 LD (PR1+1),HL
FE8C 22A6FF 00300 LD (PRCR+1),HL
FE8F 225CFF 00310 LD (PR2+1),HL
FE92 2294FF 00320 LD (PR3+1),HL
FE95 22DEFE 00330 LD (PR4+1),HL
FE98 22E3FE 00340 LD (PR5+1),HL
FE9B 22A0FF 00350 LD (PR6+1),HL
FE9E 2298FF 00360 LD (PR7+1),HL
FEA1 2183FE 00370 LD HL,ENTRY
FEA4 221640 00380 LD (KYPNTR),HL
FEA7 221144 00390 LD (MEMPRT),HL ;xxx FOR DISK ONLY xxx
FEAA 21C0FF 00400 LD HL,MSG ;POINT TO MESSAGE
FEAD CD1802 00410 CALL VOLINE ;DISPLAY IT
FE80 C32D40 00420 JP 4020H ;xxx FOR DISK ONLY xxx
00430 ;xxx LD BC,1A18H ;FOR NON-DISK SYSTEMS xxx
00440 ;xxx JP 19AEH ;LEVEL II BASIC xxx
00450 ;
FE83 3A1638 00460 ENTRY LD A,(KEYS)
FE86 FE18 00470 CP 18H ;UP & DOWN ARROWS
FE88 C24900 00480 KYDNR JP NZ,KEYSON ;EXIT IF NOT PRESSED
00490 ;PRINT THE SCREEN
FE88 F5 00500 PUSH AF ;SAVE REGISTERS
FE8C C5 00510 PUSH BC
FE8D D5 00520 PUSH DE
FE8E E5 00530 PUSH HL
FE8F 21083C 00540 LD HL,3C00H ;START OF VIDEO MEMORY
FE92 1611 00550 LD D,17 ;16+1 LINES OF THE SCREEN
FE94 15 00560 NEXTLIN DEC D
FE95 FECS CA9FF 00570 JP Z,FIN ;TERMINATE IF FINISHED
FE98 0640 00580 LD B,64 ;64 CHR. PER LINE
FE9A 4E 00590 NEXTCHR LD C,(HL) ;C CONTAINS NEXT CHR
FE9B 79 00600 LD A,C ;TRANSFER IT TO A
FE9C FE80 00610 CP 128 ;TEST FOR GRAPHICS
FE9E DA05FF 00620 JP C,CALLP ;IF NOT, PRINT AS IS
FED1 78 00630 LD A,B
FED2 FE80 00640 CP 128 ;CHECK IF FIRST GRAPH CHR
00650 ;ON THIS LINE
FED4 3802 00660 JR C,PRINCR
FED6 1800 00670 JR LAB
FED8 C680 00680 PRINCR ADD A,128
FEDA 47 00690 LD B,A ;EXTEND LINE TO 128 CHR
FEDB 0E18 00700 LD C,27 ;1/2 LINEFEED
FEDD CD0000 00710 PR1 CALL 0-4
FEE0 0E1C 00720 LD C,28 ;SECOND PART OF PRINT CTL
FEE2 CD0000 00730 PR5 CALL 0-4 ;RELOAD CHR IN C
FEE5 4E 00740 LAB LD C,(HL)
FEE6 79 00750 LD A,C
FEE7 E603 00760 AND 3 ;CLEAR ALL BIT BUT 0 & 1
FEE9 FE00 00770 CP 0 ;BIT 0 OFF?
FEEB 280A 00780 JR Z,GR32 ;BIT 0 ON?
FEED FE01 00790 CP 1
FEF0 280A 00800 JR Z,GR351 ;BIT 1 ON, 0 OFF?
FEF1 FE02 00810 CP 2
FEF3 280A 00820 JR Z,GR352 ;BIT 1 AND 0 MUST BE ON!
FEF5 180C 00830 JR GR357
FEF7 0E20 00840 GR32 LD C,32
FEF9 180A 00850 JR CALLP ;LEFT PIXEL
FEFB 0EE9 00860 GR351 LD C,233
FEFD 1806 00870 JR CALLP ;RIGHT PIXEL
FEFF 0EEA 00880 GR352 LD C,234
FF01 1802 00890 JR CALLP ;BOTH PIXEL
FF03 0EEF 00900 GR357 LD C,239
FF05 CDASFF 00910 CALLP PRCR
FF08 23 00920 INC HL
FF09 05 00930 DEC B
FF0A 78 00940 LD A,B
FF0B FE00 00950 CP 0 ;REG A IS 0 IF NO GRAPH
FF0D 2008 00960 JR NZ,NXTTST ;CHAR ON THIS LINE!
FF0F 0E0D 00970 LD C,00H ;CARRIAGE RETURN
FF11 CD0000 00980 PR1 CALL 0-4
FF14 C3C4FE 00990 JP NEXTLIN
FF17 FE80 01000 NXTTST CP 128
FF19 CA1FFF 01010 JP Z,SAME2
FF1C C3CAFE 01020 JP NTOCHR
FF1F 0E0D 01030 SAME2 LD C,00H ;REPROCESS SAME LINE
FF21 CDASFF 01040 PR8 CALL PRCR
FF24 D5 01050 PUSH DE
FF25 114000 01060 LD DE,64
FF28 ED52 01070 SEC
```



FF2A 01	01080	POP	DE	
FF2B 4E	01090 SSCND	LD	C,(HL)	
FF2C 79	01100	LD	A,C	
FF2D FEB0	01110	CP	128	
FF2F 3810	01120	JR	C,CA32	
FF31 E68C	01130	AND	12	
FF33 FE00	01140	CP	0	
FF35 280A	01150	JR	Z,CA32	
FF37 FE04	01160	CP	4	
FF39 280A	01170	JR	Z,CA351	
FF3B FE08	01180	CP	8	
FF3D 280A	01190	JR	Z,CA352	
FF3F 180C	01200	JR	CA357	
FF41 0E20	01210 CA32	LD	C,32	
FF43 180A	01220	JR	CAL0	
FF45 0EE9	01230 CA351	LD	C,233	
FF47 1806	01240	JR	CAL0	
FF49 0EEA	01250 CA352	LD	C,234	
FF4B 1802	01260	JR	CAL0	
FF4D 0EEF	01270 CA357	LD	C,239	
FF4F CDASFF	01280 CAL0	CALL	PRCR	
FF52 23	01290 SSCND	INC	HL	
FF53 05	01300	DEC	B	
FF54 78	01310	LD	A,8	
FF55 FE40	01320	CP	64	
FF57 2007	01330	JR	NZ,SSCND2	
FF59 0E0D	01340	LD	C,00H	
FF5B CD0000	01350 PR2	CALL	9-4	
FF5E 1802	01360	JR	SAME3	
FF60 18C9	01370 SSCND2	JR	SSCND	
	01380 ;			
FF62 05	01390 SAME3	PUSH	DE	;REPROCESS SAME LINE
FF63 114000	01400	LD	DE,64	
FF66 ED52	01410	SBC	HL,DE	
FF68 01	01420	POP	DE	
FF69 4E	01430 STHRD	LD	C,(HL)	
FF6A 79	01440	LD	A,C	
FF6B FEB0	01450	CP	128	
FF6D 3810	01460	JR	C,GN32	
FF6F E630	01470	AND	98	
FF71 FE00	01480	CP	0	
FF73 280A	01490	JR	Z,GN32	
FF75 FE10	01500	CP	16	
FF77 280A	01510	JR	Z,GN351	
FF79 FE20	01520	CP	32	
FF7B 280A	01530	JR	Z,GN352	
FF7D 180C	01540	JR	GN357	
FF7F 0E20	01550 GN32	LD	C,32	
FF81 180A	01560	JR	CALR	
FF83 0EE9	01570 GN351	LD	C,233	
FF85 1806	01580	JR	CALR	
FF87 0EEA	01590 GN352	LD	C,234	
FF89 1802	01600	JR	CALR	
FF8B 0EEF	01610 GN357	LD	C,239	
FF8D CDASFF	01620 CALR	CALL	PRCR	
FF90 23	01630 THRD	INC	HL	
FF91 1806	01640	DJNZ	STHRD	
FF93 0E0D	01650	LD	C,00H	
FF95 CD0000	01660 PR3	CALL	9-4	
FF98 0E1B	01670	LD	C,27	;BACK TO NORMAL LINEFEED
FF9A CD0000	01680 PR7	CALL	9-4	
FF9D 0E36	01690	LD	C,54	
FF9F CD0000	01700 PR6	CALL	9-4	
FFA2 C3C4FE	01710	JP	NXTLIN	
	01720 ;			
FFA5 CD0000	01730 PRCR	CALL	9-4	
FFA8 C9	01740	RET		
FFA9 0E0D	01750 FIN	LD	C,00H	
FFAB CDASFF	01760	CALL	PRCR	
FFAE E1	01770	POP	HL	
FFAF 01	01780	POP	DE	
FFB0 C1	01790	POP	BC	
FFB1 F1	01800	POP	AF	
FFB2 C34900	01810	JP	KEYSCN	
	01820 ;			
FFB5 F5	01830 PRDNT	PUSH	AF	
FFB6 CD4B04	01840 PRTRDY	CALL	CHPRT	;CHECK FOR PRINTER READY
FFB9 20FB	01850	JR	NZ,PRTRDY	;WAIT IF NOT
FFBB F1	01860	POP	AF	
FFBC 79	01870	LD	A,C	;LOAD CHR
FFBD D3FB	01880	OUT	(0FBH),A	;PRINT IT

FFBF C9	01890	RET	
FFC8 53	01900 MSG	DEFB	'Screen print installed, (by S. Calmettes, 613-3843729)
	63 72 65 65 6E 20 70 72 69 6E 74 20 69 6E 73 74		
	61 6C 6C 65 64 2C 20 28 62 79 20 53 2E 20 43 61		
	6C 6D 65 74 74 65 73 2C 20 36 31 33 2D 33 38 34		
	33 37 32 39 29		
FFF6 0D	01910	DEFB	00H
FEB0	01920	END	SETUP
00000	TOTAL ERRORS		

  

CALLP	FF05	CAL0	FF4F	CALR	FF8D	CHPRT	040B	ENTRY	FEB3
FIN	FFA9	CA32	FF41	CA351	FF45	CA352	FF49	CA357	FF4D
GN32	FF7F	GN351	FF83	GN352	FF87	GN357	FF8B	GN32	FEF7
GN351	FEFB	GN352	FEFF	GN357	FF83	KEYS	3840	KEYSCN	0049
KYDAR	FEB8	KYPNTR	4016	LAB	FEE5	MEMPR	4411	MSG	FFC0
NXTCHR	FEC4	NXTLIN	FEC4	NXTTST	FF17	PR1	FF11	PR2	FF5B
PR3	FF95	PR4	FED0	PR5	FEE2	PR6	FF9F	PR7	FF9A
PR8	FF21	PROR	FFA5	PRIMER	FED8	PRINT	FFB5	PRTRDY	FFB6
SAME2	FF1F	SAME3	FF62	SCND	FF32	SCND2	FF60	SETUP	FEB0
SSCND	FF2B	STHRD	FF69	THRD	FF90	VOLINE	021B		

## LISTING 2

```

10 'SCREEN PRINT (SCRPT)
20 'BY SERGE Y. CALMETTES,
30 'FOR THE TRS-80 MODEL III, TO PRINT THE CONTENTS OF THE
40 'SCREEN, INCLUDING GRAPHIC CHARACTERS.
50 'LEVEL II BASIC VERSION FOR 16K
60 FORI=32384TO32758
70 READA:POKEI,A
80 NEXTI
90 INPUT"Press enter to activate SCREEN PRINT";EN
100 POKE16526,128:POKE16527,126:X=USR(Y)
110 DATA 042,022,064,034,185,126,033,181,127,034,018,127,034
120 DATA 166,127,034,092,127,034,150,127,034,222,126,034
130 DATA 227,126,034,160,127,034,155,127,033,179,126,034
140 DATA 022,064,033,192,127,205,027,002,001,024,026,195
150 DATA 174,025,058,064,056,254,024,194,073,000,245,197
160 DATA 213,229,033,000,060,022,017,021,202,169,127,006,064
170 DATA 078,121,254,128,218,005,127,120,254,128,056,002
180 DATA 024,013,198,128,071,014,027,205,000,000,014,028,205
190 DATA 000,000,078,121,230,003,254,000,040,010,254,001,040
200 DATA 010,254,002,040,010,024,012,014,032,024,010,014,233
210 DATA 024,006,014,234,024,002,014,239,205,165,127,035,005
220 DATA 120,254,000,032,008,014,013,205,000,000,195,196,126
230 DATA 254,128,202,031,127,195,202,126,014,013,205,165,127
240 DATA 213,017,064,000,237,082,209,078,121,254,128,056,016
250 DATA 230,012,254,000,040,010,254,004,040,010,254,008,040
260 DATA 010,024,012,014,032,024,010,014,233,024,006,014,234
270 DATA 024,002,014,239,205,165,127,035,005,120,254,064,032
280 DATA 007,014,013,205,000,000,024,002,024,201,213,017,064
290 DATA 000,237,082,209,078,121,254,128,056,016,230,048,254
300 DATA 000,040,010,254,016,040,010,254,032,040,010,024,012
310 DATA 014,032,024,010,014,233,024,006,014,234,024,002,014
320 DATA 239,205,165,127,035,016,214,014,013,205,000,000,014
330 DATA 027,205,000,000,014,054,205,000,000,195,196,126,205
340 DATA 000,000,201,014,013,205,165,127,225,209,193,241
350 DATA 195,073,000,245,205,075,004,032,251,241,121,211,248
360 DATA 201,083,099,114,101,101,110,032,112,114,105,110,116
370 DATA 032,105,110,115,116,097,108,108,101,100,044,032,040
380 DATA 098,121,032,083,046,032,067,097,108,109,101,116
390 DATA 116,101,115,044,032,054,049,051,045,057,054,054,049
400 DATA 053,048,054,041,013

```

## TIPS FOR VIDEOGENIE (PMC-80, SYSTEM-80)

by H. Delahaye, Veenendaal, Holland

Translated by Paul Fransen

When using a Videogenie the printer is port addressed. That's why you can't get a hardcopy with the Tandy EDTASM program. You can fix this by making the following changes:

Load the system tape. Don't start the program but instead press BREAK. Then change the following bytes:

POKE 19002,253	HEX: 4A3A,FD
POKE 19029,253	4A55,FD
POKE 19082,0	4A8A,00
POKE 19083,219	4A8B,DB
POKE 19084,253	4A8C,FD

Then start again by entering SYSTEM. When you see the 'e?' enter '/' and ENTER.

With the right Debug program you can save the changed program on tape.

# BASIC WORD SEARCH PROGRAM

by Jim Mumaugh

I was pleased to see that you were interested in some BASIC language programs for the Northern Bytes. I have written a few and will send them along. I thought your readers were more interested in Assembly so I never bothered to ask before.

This name of this program is WRDSRCH/BAS. I altered a program a teacher at our high school had. Several teachers wanted a wordsearch program which would intertwine the words. When I got this program, it also did not mix the words. I don't know who those other guys are, but I left their names in anyway because I used their basic ideas. The actual program has little resemblance to the original.

## FEATURES OF WRDSRCH/BAS

You are not limited as to the number of words you insert. Of course, the total letter count needs to be about 100 less than available space in order to get the words into the puzzle.

You are limited to 40 characters wide because of the standard 80 character width of most printers. The length should be greater than 20 but there is no actual limit.

Occasionally you will be unable to fit all words into a puzzle. You can request the program to retry as many times as you like but somewhere there will be a situation when one or two words don't fit. I have planned for that in lines 19622 and 19624. This way a teacher could have a word needed "searched for" by the student even if it was not actually in the puzzle.

I included some data to use as a demonstration. It's easier when showing someone if you don't have to think up 40 or 50 words before you show it off.

I added some entertainment while the program worked, while keeping the user abreast of the current status of the program. Everybody seems to enjoy the comments.

The most important items I added were the "save to disk" routines. Once you type in a word list, you can save and recall it. This is especially useful when a teacher wants two or more puzzles with the same words. I also offer to write the puzzle and its solution to disk (or to the printer). I prefer the disk save because I only need to keep the disk version and can print it on a ditto master when needed using PRINT filename/PZL from DOS ready.

I think the solution is actually worthless. I used it while debugging the program to be sure it would interlock the words. If you get a ratio of used spaces to available spaces near 1 you will have most words interlocked in 2 or more spaces. I recommend than you press break when I announce that I am ready to write the solution to disk. The program is over at this point and the solution just takes up disk space.

The actual interlocking is random. Sometimes you will get lots of it and other times only a few. I simply pick a random spot in the grid to begin my word. Then I look at each letter in my word and compare it with the next space. If there is no conflict I continue. (The same letter as I am currently looking at or a blank is OK.) If there is a conflict, I retry at another location in the same direction. I try 20 times then pick a new direction. If after 160 tries I still don't get it in, I say it's impossible but offer to try another 160 times. Many times I have tried over 300 times and then it would fit, so have patience.

One last feature I added (because many of the teachers using this program are not more than very casual users) is that I pick the puzzle name, solution name and word list names for them. You do have the option to override these, however, when you use two puzzles with the same first word in their titles.

I hope you find this interesting and useful.

Jim Mumaugh  
J & J Computer Consultant  
(also Vacaville High School Advanced Mathematics teacher)  
244 North Alamo Drive  
Vacaville, California 95688

```
1700 CLEAR10000:CLS:RANDOM:DEFINT A-
Z:FOR I=1 TO 10:READ EX$(I):NEXT
1720 DATA Zounds!, Ah Ha!, Yippee!, Umph!, Aargh!, Finally!, Whew!, Of co
urse!, Can't fool me!, Naturally
1800 '-----SET UP MATRIX-----
1840 PRINT "Do you want a demonstration? Y/N ";:LINE INPUT Y$:IF
LEFT$(Y$,1) = "Y" THEN DIM A(26,21), D$(50):A1=25:B1=20:NA$="DEMON
STRATION":WN=50:FOR I=1 TO WN:READ D$(I):NEXT I:GOTO 2700
1850 INPUT "ENTER PUZZLE NAME":NA$
1860 INPUT "Do you have a word list ALREADY on disk?";W$:IF W$="" THEN 1900
1870 INPUT "What file name does it have?";F2$:OPEN "I",1,F2$:INPUT #
1,WN:DIM D$(WN):FOR I=1 TO WN:INPUT #1,D$(I):NEXT I:CLOSE
1900 PRINT "What size of matrix would you like? I recommend a 20
X 20 with about 40 words Enter two numbers, separated with co
mmas.
1905 INPUT "What are your two numbers?";A1,B1
2000 IF A1>40 THEN 1900:IF B1<1/2 LENGTH OF LINE PRINTER
2100 DIM A(A1+1,B1+1)
2200 '-----ENTER NUMBER OF WORDS-----
2300 D=INT(A1*B1/10)
2305 IF W$="Y" THEN 2700
2310 PRINT "You have room for about "D" words (give or take 10). Ho
w many words shall we use?";:INPUT WN
2500 DIM D$(WN)
2600 FOR Q=1 TO WN:PRINT "ENTER WORD NUMBER";Q:INPUT D$(Q):
NEXT Q
2700 GOSUB 9900:Q=0:CLS:GOSUB 11000
2705 INPUT "Shall I save this word list to disk?";W$:IF W$="" THEN 2
800
2720 IF VAL(NA$) < 0 THEN NN$="Z"+NA$ ELSE NN$=NA$
2730 IF LEN(NN$)>8 THEN NN$=LEFT$(NN$,8)
2740 X=INSTR(NN$,"") IF X=0 THEN X=8
2750 WR$=LEFT$(NN$,X-
1)+"/WRD":PRINT "The word list will be called "WR$"."
2760 PRINT "Is this satisfactory with you? ";:LINE INPUT Y$:IF Y$=""
THEN 2765
2763 INPUT "What filename shall I use?";WR$
2765 OPEN "O",1,WR$:PRINT #1,WN,"";:FOR I=1 TO WN:PRINT #1,D$(I)
,"";:NEXT I:CLOSE
2800 TY=0
2810 Q=Q+1:A$=D$(Q)
2820 PRINT "I am working on : "A$".";
2900 '-----PICK DIRECTION-----
3000 DI=RND(8)
3010 ON DI GOTO 3300,3800,4300,4800,5300,5900,6500,7100
3100 '-----SET UP ARRAY-----
3200 ' FORWARD FIXED/JIM
3300 F1=0:A=RND(A1):B=RND(B1):IF A+LEN(A$)>A1 THEN 3300
3400 I=0:FOR T=A TO A+LEN(A$)-1
3405 I=I+1
3410 B$=MID$(A$,I,1):IF (A(T,B)=0 OR ASC(B$)=A(T,B)) THEN 3420
3415 T=A+LEN(A$):F1=1
3420 NEXT T
3430 IF F1=1 THEN 10800
3500 FOR T=1 TO LEN(A$):B$=MID$(A$,T,1):A(A,B)=ASC(B$)
3600 A=A+1:NEXT T:GOTO 7600
3700 ' BACKWARD FIXED/JIM
3800 F1=0:A=RND(A1):B=RND(B1):IF A-LEN(A$)<0 THEN 3800
3900 I1=LEN(A$)+1:I=0:FOR T=A-LEN(A$)+1 TO A
3905 I=I+1
3910 B$=MID$(A$,I1-1,1):IF (A(T,B)=0 OR ASC(B$)=A(T,B)) THEN 3920
3915 T=A:F1=1
3920 NEXT T
3930 IF F1=1 THEN 10800
4000 FOR T=1 TO LEN(A$):B$=MID$(A$,T,1):A(A,B)=ASC(B$)
4100 A=A-1:NEXT T:GOTO 7600
4200 ' DOWN FIXED/JIM
4300 F1=0:A=RND(A1):B=RND(B1):IF B+LEN(A$)>B1 THEN 4300
4400 I=0:FOR T=B TO B+LEN(A$)-1
4405 I=I+1
4410 B$=MID$(A$,I,1):IF (A(A,T)=0 OR ASC(B$)=A(A,T)) THEN 4420
4415 T=B+LEN(A$):F1=1
4420 NEXT T
4430 IF F1=1 THEN 10800
4500 FOR T=1 TO LEN(A$):B$=MID$(A$,T,1):A(A,B)=ASC(B$)
4600 B=B+1:NEXT T:GOTO 7600
4700 ' UP FIXED/JIM
```

```
1000 ' *****
1050 ' 12/17/84 6:05 PM *
1100 ' WRDSRCH/BAS *
1150 ' *
1200 ' WRITTEN BY RON BENNINGHOF *
1250 ' MODIFIED BY JIM LARISON (9 FATAL ERRORS) *
1300 ' REWRITTEN BY JIM MUMAUGH *
1400 ' *
1450 ' *****
1500 '
1550 '
```

```

4800 F1=0:A=RND(A1):B=RND(B1):IFB-LEN(A$)<0THEN4800
4900 I=0:I1=LEN(A$)+1:FORT=B-LEN(A$)+1TOB
4905 I=I+1
4910 B$=MID$(A$,I1-I,1):IF (A(A,T)=0 OR ASC(B$)=A(A,T))THEN4920
4915 T=B:F1=1
4920 NEXT T
4930 IF F1=1 THEN 10800
5000 FORT=1TOLEN(A$):B$=MID$(A$,T,1):A(A,B)=ASC(B$)
5100 B=B-1:NEXTT:GOTO7600
5200 ' DOWN DIAGONALLY
5300 F1=0:A=RND(A1):B=RND(B1):IFA+LEN(A$)>A1THEN5300
5400 IFB+LEN(A$)>B1THEN5300
5500 I=0:FORT=0TOLEN(A$)-1
5505 I=I+1
5510 B$=MID$(A$,I,1):IF (A(A+T,B+T)=0 OR ASC(B$)=A(A+T,B+T)) TH
EN5520
5515 T=LEN(A$):F1=1
5520 NEXT T
5530 IF F1=1 THEN 10800
5600 FORT=1TOLEN(A$):B$=MID$(A$,T,1):A(A,B)=ASC(B$)
5700 A=A+1:B=B-1:NEXT T:GOTO7600
5800 ' UP DIAGONALLY
5900 F1=0:A=RND(A1):B=RND(B1):IFA+LEN(A$)>A1 THEN 5900
6000 IF B-LEN(A$)<1 THEN 5900
6100 I=0:FOR T=0 TO LEN(A$)-1
6105 I=I+1
6110 B$=MID$(A$,I,1):IF (A(A+T,B-T)=0 OR ASC(B$)=A(A+T,B-
T))THEN6120
6115 T=LEN(A$):F1=1
6120 NEXT T
6130 IF F1=1 THEN 10800
6200 FOR T=1 TO LEN(A$):B$=MID$(A$,T,1):A(A,B)=ASC(B$)
6300 A=A+1:B=B-1:NEXT T:GOTO7600
6400 ' UP DIAGONALLY
6500 F1=0:A=RND(A1):B=RND(B1):IFA-LEN(A$)<1THEN 6500
6600 IF B-LEN(A$)<1 THEN 6500
6700 I=0:FOR T=0 TO LEN(A$)-1
6705 I=I+1
6710 B$=MID$(A$,I,1):IF (A(A-T,B-T)=0 OR ASC(B$)=A(A-T,B-
T)) THEN 6720
6715 T=LEN(A$):F1=1
6720 NEXT T
6730 IF F1=1 THEN 10800
6800 FOR T=1 TO LEN(A$):B$=MID$(A$,T,1):A(A,B)=ASC(B$)
6900 A=A-1:B=B+1:NEXT T:GOTO7600
7000 ' DOWN DIAGONALLY
7100 F1=0:A=RND(A1):B=RND(B1):IFA-LEN(A$)<1THEN 7100
7200 IF B+LEN(A$)>B1 THEN 7100
7300 I=0:FOR T=0 TO LEN(A$)-1
7305 I=I+1
7310 B$=MID$(A$,I,1):IF (A(A-T,B+T)=0 OR ASC(B$)=A(A-
T,B+T))THEN7320
7315 T=LEN(A$):F1=1
7320 NEXT T
7330 IF F1=1THEN10800
7400 FOR T=1 TO LEN(A$):B$=MID$(A$,T,1):A(A,B)=ASC(B$)
7500 A=A-1:B=B+1:NEXT T
7600 IF Q<WN THEN PRINTEX$(RND(10))" I got it."!GOTO2800 ELS
E PRINT"I got the last one!"!TB=(80-A1*2)/2:CMD"O",WN,D$(1)
7700 '-----PRINT OUT MATRIX-----
7705 PRINT"Now that that job is finished, I need to know if you want
me to save this puzzle to disk or write directly to the printer."!PRIN
T"If you want to write it to DISK then type 'DISK' otherwise type 'P
RINTER' ==> "!!INPUT CH$
7710 IF CH$="DISK" THEN GOTO 17700 ELSE IF CH$<>"PRINTER" T
HEN GOTO7705
7750 Q=(80-
LEN(NA$))/2:PRINTTAB(Q)NA$:PRINT:PRINT"PRINTING PUZZLE
NOW"
7800 FOR B=1 TO B1:FOR A=1 TO A1:PRINTTAB(TB);
7900 IF A(A,B)=0 THEN LPRINT CHR$(64+RND(26)):GOTO8100
8000 LPRINTCHR$(A(A,B));
8100 LPRINT" ";NEXT A:LPRINT:LPRINTTAB(TB);NEXT B:GOSUB96
20
9000 '-----GIVE SOLUTION-----
9010 PRINT"Press <ENTER> to print the solution";LINEINPUTY$
9100 PRINT:PRINT"NOW FOR THE SOLUTION..."
9250 LPRINTNA$+"! SOLUTION"!LPRINT
9300 FOR B=1 TO B1:FOR A=1 TO A1:PRINTTAB(TB);

```

```

9400 IF A(A,B)=0 THEN LPRINT".";GOTO9600
9500 LPRINTCHR$(A(A,B));
9600 LPRINT" ";NEXT A:LPRINT:LPRINTTAB(TB);NEXT B:GOSUB96
20:GOTO9700
9610 '-----PRINT OUT THE WORDS-----
9620 IF WR=0GOTO9630
9622 LPRINT:LPRINT"Only"WR"of these words are NOT in the puzzle
."!
9624 IFWR=1LPRINT"Which word is it?"ELSELPRINT"Which words ar
e they?"
9630 LPRINT:LPRINTTAB(5);A=0:FOR Q=1 TO WN:A=A+1
9632 IF A=5 LPRINT:LPRINTTAB(5);A=1
9640 LPRINT D$(Q);IF A<4 LPRINTSTRING$(20-LEN(D$(Q))," ");
9650 NEXT Q:LPRINT:LPRINT:PRINT:PRINT:RETURN
9660 '-----SHALL WE QUIT?-----
9700 INPUT"ANOTHER COPY OF SAME PUZZLE";OO$
9710 IF LEFT$(OO$,1)="Y"THEN 7700 ELSE END
9800 '-----EDIT SUBROUTINE-----
9900 CLS:FOR Q=1 TO WN:PRINT Q;"="!D$(Q);NEXT Q
10000 PRINT:INPUT"DO YOU WANT TO CHANGE ANY WORDS (Y OR
N)";D$
10100 D1$=LEFT$(D$,1):IF D1$="N" THEN RETURN
10200 INPUT"WHICH WORD DO YOU WANT TO CHANGE";Q
10300 INPUT"TYPE THE NEW WORD ";D$(Q):GOTO9900
10700 '-----ATTEMPTED TRIES-----
10800 TY=TY+1
10810 IF TY/160=INT(TY/160) THEN PRINT"I think this one's impos
sible! Shall I try again? Y/N "!!INPUTY$:IFY$="N"THENWR=WR+
1:GOTO2810ELSE2820
10820 IF TY/20=INT(TY/20) THEN PRINT"It wouldn't fit."!GOTO300
0
10825 GOTO3010
11000 '-----SORT FROM LONGEST TO SHORTEST WORD
11010 FOR I=1 TO WN
11020 D$(I)=STR$(100+LEN(D$(I)))+D$(I)
11030 NEXT
11040 CMD"O",WN,-D$(1)
11050 FOR I=1 TO WN
11060 D$(I)=RIGHT$(D$(I),LEN(D$(I))-4)
11070 NEXT
11080 RETURN
11090 '
17700 '-----WRITE TO DISK-----
17710 IFVAL(NA$)>0THENNN$="Z"+NA$ELSEN$=NA$
17712 IF LEN(NN$)>8THENNN$=LEFT$(NN$,8)
17715 X=INSTR(NN$,"")!IFX=0THENX=8
17718 PZ$=LEFT$(NN$,X-1)+"/PZL"!SL$=LEFT$(NN$,X-
1)+"/SOL"!PRINT"The puzzle will be called "PZ$" and the solution wil
l be called "SL$".
17719 PRINT"Is this satisfactory with you? "!!LINEINPUTY1$:IFY1
$<>"N"THEN17745
17730 INPUT"Ok. What name would you like";N$
17735 IF LEN(NN$)>8 THEN PRINT"Eight letters and/or numbers, plea
se."!GOTO17730
17736 X=INSTR(NN$,"/"):IF X<0 THENPRINT"I will supply the extens
ion."!GOTO17730
17737 IF VAL(NN$)<0 THEN PRINT"The name must begin with a lette
r."!GOTO17730
17740 GOTO17715
17745 OPEN"O",1,PZ$
17750 Q=(80-
LEN(NA$))/2:PRINT#1,TAB(Q)NA$:PRINT#1:PRINT"PRINTING PUZ
ZLE NOW"
17800 FOR B=1 TO B1:FOR A=1 TO A1:PRINT#1,TAB(TB);
17900 IF A(A,B)=0 THEN PRINT#1,CHR$(64+RND(26)):GOTO18100
18000 PRINT#1,CHR$(A(A,B));
18100 PRINT#1," ";NEXT A:PRINT#1:PRINT#1,TAB(TB);NEXT B:G
OSUB19620:CLOSE
19000 '-----GIVE SOLUTION-----
19010 PRINT"Press <ENTER> to print the solution";LINEINPUTY$
19100 PRINT:PRINT"NOW FOR THE SOLUTION..."
19200 OPEN"O",1,SL$
19250 PRINT#1,NA$+"! SOLUTION"!PRINT#1,
19300 FOR B=1 TO B1:FOR A=1 TO A1:PRINT#1,TAB(TB);
19400 IF A(A,B)=0 THEN PRINT#1,"";GOTO19600

```

```

19300 PRINT#1,CHR$(A,B));
19600 PRINT#1,"";NEXT A:PRINT#1,PRINT#1,TAB(TB);NEXT B:G
OSUB19620:CLOSE:END
19610 '-----PRINT OUT THE WORDS-----

```

```

19620 IF WR=0GOTO19630
19622 PRINT"There were"WR"words NOT included in the puzzle.";P
RINT#1,"";PRINT#1,"Only"WR"of these words are NOT in the puzzle.";
19624 IFWR=1PRINT#1," Which word is it?"ELSEPRINT#1," Which w
ords are they?"
19630 PRINT#1,PRINT#1,TAB(5);A=0;FOR Q=1 TO WN:A=A+1
19632 IF A=5 PRINT#1,PRINT#1,TAB(5);A=1
19640 PRINT#1, D$(Q);IF A<4 PRINT#1,STRING$(20-
LEN(D$(Q))," ");
19650 NEXT Q:PRINT#1,PRINT#1,PRINT:PRINT:RETURN
20001 DATA MUMAUGH,SANFRANCISCO,CLEVELAND,MIAMI,PHILAD
ELPHIA,NEWENGLAND,BUFFALO,ATLANTA,NEWORLEANS,KANSAS
CITY,HOUSTON,GREENBAY,MINNESOTA,NYJETS,INDIANAPOLIS,ST
LOUIS,DALLAS,WASHINGTON,DETROIT
20002 DATA PITTSBURG,CINCINNATI,NYGIANTS,TAMPABAY,DENV
ER,SANDIEGO,CHICAGO,LARAMS,SEATTLE,LARADERS,FOOTBALL
20003 DATA IOWA,YALE,ARMY,NAVY,OHIOST,ARIZONA,OREGON,R
ICE,BAYLOR,STANFORD,CORNELL,BROWN,CLEMSON,TEXAS,KENTU
CKY,FLORIDA,PURDUE,GEORGIA,COLLEGE,RUTGERS

```

```

*****

```

#### OVERSEAS EXPERIENCE by "Computer Nut" (Part 4 of a series)

Once you have purchased the software and hardware you require, then comes the question of support. Support should never be more than a phone call away, but a phone call from another country can be an expensive business, even costing more than the software itself.

I know software publishers get calls and letters from all sorts of people, with questions ranging from which way to orient the disk when mounting it in the drive, to why printer I using buffer J and filter K on operating system L (no, not necessarily 'elldos) using computer M drops the first character after a form feed spacing more than N lines. It can be a frustrating job answering all these questions, especially those which are already answered within the first few pages of the documentation.

It should be important, both to yourself and to the publisher, that you fill in the registration card and send it off when you have bought some software. Some publishers apparently throw registration cards in the nearest bin, and you never hear another word from them, while others use the information for sending newsletters and notice of new updates to programs. I don't expect publishers to support users who do not bother to register their purchase, or users who have not obtained their copy of the program legitimately.

I do, however, expect publishers to answer genuine and legitimate questions from registered users concerning their products, whether it be by phone or by letter. A software publisher who does not bother to answer my questions or even acknowledge the receipt of them cannot expect to sell me any more products, or expect me to recommend the product to others. Despite this, several of them seem to survive or even thrive.

I also expect publishers to inform me when they upgrade their products, and provide me with the possibility of upgrading at a reasonable price. One of the larger software publishers, no longer in the TRS-80 business, provide an update possibility within the same version number, but require a new purchase for major revisions of the product. I bought one of their products about 2 months before a major update, and now I have to purchase the product again at full price if I want the new features. They probably won't notice it, but if I can avoid buying any of their products in future I will do so. Also, since their license agreement bound them to an upgrade policy and since they have not kept their part of the agreement, I no longer feel bound by my signature on the agreement.

Another feature which I appreciate from publishers is a newsletter. This could be provided free of charge for a certain period after a software purchase, or at a nominal fee. Newsletters can provide useful information such as corrections to known bugs, restrictions, feedback from users, notice of upgrades, and useful information which was not included in the manual. The newsletter does not have to be a large, bound volume. One or two photocopied sheets would be sufficient for most products.

For overseas users Tandy/Radio Shack has a lot to learn about user support. Apparently they assume that all users have ready access to a local store where update notices and correction reports can be read at leisure. At the moment I don't even know if there is a Tandy store in the country where I live, let alone in the same area. I have registered several Tandy software products, but only after direct correspondence concerning serious bugs have I received the patches required. I have no idea what they use the registration cards for, since I have never received any mail from them except after writing and asking for information.

I don't know if they expect me to write every few months asking if there are any updates to their products, or how they expect me to get the disks containing new releases. Until now I have been fortunate enough to have friends who traveled regularly to the U.S. who could get copies as and when required, but surely they can't expect all users to be in that situation.

For the time being this will be the last installment in this series. If you have any comments or suggestions please send them to me care of Jack Decker. The address is on the back of this issue of Northern Bytes.

```

*****

```

#### SZAP80/CMD A program to patch Model III SUPERZAP to work with VIDEO4 by Tony Domigan

If you own a Model 4 or 4P and use a program such as VIDEO4 to enable the 24x80 screen display while in the Model III mode, you may find this program useful. It patches the SUPERZAP program of NEWDOS/80 to permit operation in either the normal 64x16 or in the 80x24 mode. Although this patched version of SUPERZAP is 100H longer than the original, it is entirely self-contained. To use it normally, you simply enter 'SUPERZAP' as usual; but if you wish to use it in the 80x24 mode, you enter 'SUPERZAP 4'.

The LDAHL, LDHLA and CONVRT routines have been borrowed (with permission) from VIDEO4. Further instructions for installation and operation may be found in the source code comments below.

```

00010 ; SZAP80/ASM - Model 4
00020 ; 80x24 mode for Superzap
00030 ; For Northern Bytes & the Public Domain
00040 ; by Tony Domigan
00050 ; P.O. Box 150, Thornestown, Victoria, 3074, AUSTRALIA.
00060 ; MCI-ID Z54-5121
00070 ;
00080 ; Instructions to create a dual version (64x16 and 80x24)
00090 ; 1. From DOS 'LOAD SUPERZAP/CMD'
00100 ; 2. From DOS 'LOAD SZAP80/CMD'
00110 ; 3. From DOS 'DUMP SUPERZAP/CMD 5200H,6FFFH,5300H'
00120 ; To execute -
00130 ; Enter 'SUPERZAP' for normal SUPERZAP
00140 ; Enter 'SUPERZAP 4' for 80x24 SUPERZAP
00150 ;
00160 ; This patch has been tested with VIDEO4 & VIDEO4x24
00170 ; -----
00180 ; Throw-away code - placed in SUPERZAP buffer
00190 ; Checks for '4' and executes patch if so.
00191 ; -----
5300 00200 ORG 5300H ;SUPERZAP buffer
5300 7E 00210 CHECK LD A,(HL) ;'SUPERZAP 4<cr>' ?
5301 FE34 00220 CP '4' ;80x24 patch?
5303 CA0953 00230 JP Z,START ;Yes, Execute Patch
5306 C3C354 00240 JP 54C3H ;Emc Superzap
5309 3EC3 00250 START LD A,0C3H ;Place Jumps
530B 32D656 00260 LD (56D6H),A
530E 32D85D 00270 LD (5D08H),A
5311 321560 00280 LD (6015H),A
5314 321E60 00290 LD (601EH),A
5317 320B62 00300 LD (620BH),A
531A 3EC0 00310 LD A,0C0H ;Place Calls
531C 32D856 00320 LD (56D8H),A
531F 32F3F 00330 LD (5FF3H),A
5322 323560 00340 LD (6035H),A
5325 323E60 00350 LD (603EH),A
5328 325061 00360 LD (6150H),A
532B 3E34 00370 LD A,B0 ;80 Columns
532D 32FE60 00380 LD (60FEH),A
5330 32105E 00390 LD (5E10H),A
5333 87 00400 ADD A,A ;2x80 Column Lines

```

5331	32F74F	00410	LD	(40F7H),A	
5337	21204F	00420	LD	HL,RTNE1	;Place Addresses
533A	221F4F	00430	LD	(401FH),HL	
533D	21204F	00440	LD	HL,RTNE2	
5340	22F45F	00450	LD	(3FF4H),HL	
5343	21334F	00460	LD	HL,RTNE3	
5346	220C42	00470	LD	(420CH),HL	
5349	21384F	00480	LD	HL,RTNE4	
534C	22164F	00490	LD	(4016H),HL	
534F	213F4F	00500	LD	HL,RTNE5	
5352	22364F	00510	LD	(4036H),HL	
5355	21464F	00520	LD	HL,RTNE6	
5358	223F4F	00530	LD	(403FH),HL	
535B	21524F	00540	LD	HL,RTNE7	
535E	220C5D	00550	LD	(3DDCH),HL	
5361	218D4F	00560	LD	HL,RTNE8	
5364	225141	00570	LD	(4151H),HL	
5367	21934F	00580	LD	HL,RTNE9	
536A	220C56	00590	LD	(56DCH),HL	
536D	21984F	00600	LD	HL,RTNE0	
5370	22D756	00610	LD	(56D7H),HL	
5373	21403D	00620	LD	HL,3D40H	
5376	22B660	00630	LD	(40B6H),HL	
5379	21803E	00640	LD	HL,3EB0H	
537C	22D860	00650	LD	(40D8H),HL	
537F	21C03F	00660	LD	HL,3FC0H	
5382	22E860	00670	LD	(40E8H),HL	
5385	21B140	00680	LD	HL,40B1H	
5388	22F160	00690	LD	(40F1H),HL	
538B	C3C554	00700	EXEC JP	54C5H	;Start SUPERZAP
;-----					
6F20		00720	ORG	6F20H	;Patch Area
6F20 E1		00730	RTNE1 POP	HL	
6F21 D5		00740	PUSH	DE	
6F22 111000		00750	LD	DE,16	;+64-80 Columns
6F25 19		00760	ADD	HL,DE	;New Pointer
6F26 08		00770	EX	AF,AF'	
6F27 D1		00780	POP	DE	
6F28 3D		00790	DEC	A	
6F29 C32160		00800	JP	6021H	
6F2C D5		00810	RTNE2 PUSH	DE	
6F2D 112800		00820	LD	DE,0028H	;Offset to Ascii Side
6F30 19		00830	ADD	HL,DE	
6F31 D1		00840	POP	DE	
6F32 C9		00850	RET		
6F33 C0AE6F		00860	RTNE3 CALL	LDHLA	;LD (HL),A
6F36 23		00870	INC	HL	
6F37 C9		00880	RET		
6F38 C0336F		00890	RTNE4 CALL	RTNE3	;LD (HL),A
6F3B E3		00900	EX	(SP),HL	
6F3C C31860		00910	JP	6018H	
6F3F F5		00920	RTNE5 PUSH	AF	
6F40 7B		00930	LD	A,E	
6F41 C0336F		00940	CALL	RTNE3	;LD (HL),A
6F44 7A		00950	LD	A,D	
6F45 C0AE6F		00960	CALL	LDHLA	;LD (HL),A
6F48 F1		00970	POP	AF	
6F49 C9		00980	RET		
6F4A 3E14		00990	RTNE6 LD	A,14H	;Page 1 of Video
6F4C D384		01000	OUT	(04H),A	;Select Page 1
6F4E 11003C		01010	LD	DE,3C00H	;Top of page
6F51 C9		01020	RET		
6F52 D5		01030	RTNE7 PUSH	DE	
6F53 C5		01040	PUSH	BC	
6F54 11003C		01050	LD	DE,3C00H	
6F57 19		01060	ADD	HL,DE	;Mod/Find Posn
6F58 E5		01070	PUSH	HL	
6F59 11003C		01080	LD	DE,3C00H	;Start of Screen
6F5C ED52		01090	SBC	HL,DE	;Displacement
6F5E 3E40		01100	LD	A,64	;Normal Columns
6F60 C0594C		01110	CALL	4C59H	;Calc Number of rows
6F63 F5		01120	PUSH	AF	;save remainder
6F64 3E50		01130	LD	A,80	;New Columns
6F66 C0374C		01140	CALL	4C37H	;New Displacement
6F69 F1		01150	POP	AF	;Restore Old Rows
6F6A 5F		01160	LD	E,A	;DE=Start+Remainder
6F6B 163C		01170	LD	D,3CH	
6F6D 19		01180	ADD	HL,DE	;80x24 Screen Pos
6F6E C0A76F		01190	CALL	LDHLA	;LD A,(HL)
6F71 4F		01200	LD	C,A	;Save screen char
6F72 3AE15D		01210	PLACIT LD	A,(5DE1H)	;Store Mod/Find Char

6F75 C0AE6F	01220	CALL	LDHLA	;LD (HL),A
6F78 AF	01230	XOR	A	
6F79 C0E15D	01240	CALL	5DECH	;HSCAN
6F7C F5	01250	PUSH	AF	
6F7D 79	01260	LD	A,C	;Restore screen char
6F7E C0AE6F	01270	CALL	LDHLA	;LD (HL),A
6F81 F1	01280	POP	AF	
6F82 C0E15D	01290	CALL	Z,5DECH	
6F85 28EB	01300	JR	Z,PLACIT	;No Key
6F87 E1	01310	POP	HL	
6F88 C1	01320	POP	BC	
6F89 D1	01330	POP	DE	
6F8A C0E15D	01340	JP	5DECH	
6F8D C0A76F	01350	CALL	LDHLA	;LD A,(HL)
6F90 FE20	01360	CP	20H	
6F92 C9	01370	RET		
6F93 3E14	01380	LD	A,14H	;Select vid page 1
6F95 D384	01390	OUT	(04H),A	
6F97 C01462	01400	CALL	6214H	
6F9A C9	01410	RET		
6F9B C0E15D	01420	CALL	5DECH	;Mod/Find Rtn
6F9E F5	01430	PUSH	AF	
6FA0 3E14	01440	LD	A,14H	;Select vid page 1
6FA1 D384	01450	OUT	(04H),A	
6FA3 F1	01460	POP	AF	
6FA4 C0E15D	01470	JP	5DECH	

The following Routine was taken from  
VIDEO4 by Jack Decker (avail from TAS)

6FA7 E5	01530	LDHLA	PUSH	HL	;LD A,(HL) rtn
6FAB C0B76F	01540	CALL	CONV		
6FAB 7E	01550	LD	A,(HL)		
6FAC E1	01560	POP	HL		
6FAD C9	01570	RET			
6FAE E5	01580	LDHLA	PUSH	HL	;LD (HL),A rtn
6FAF F5	01590	PUSH	AF		
6FB0 C0B76F	01600	CALL	CONV		
6FB3 F1	01610	POP	AF		
6FB4 77	01620	LD	(HL),A		
6FB5 E1	01630	POP	HL		
6FB6 C9	01640	RET			
6FB7 7C	01650	CONV	LD	A,H	;Screen Ptr Conversion
6FB8 E603	01660	AND	03H		
6FBA F63C	01670	OR	3CH		
6FBC CB74	01680	BIT	6,H		;Page 1
6FBE 67	01690	LD	H,A		
6FBF 3ACE6F	01700	LD	A,(FLAG)		
6FC2 CBFF	01710	SET	7,A		;Page 2
6FC4 2002	01720	JR	NZ,LOWER		
6FC6 E67F	01730	AND	7FH		
6FC8 32CE6F	01740	LOWER	LD	(FLAG),A	;Select Page
6FCB D384	01750	OUT	(04H),A		
6FCD C9	01760	RET			
6FCE 05	01770	FLAG	DEFB	05H	;Video Mask
0000	01780	END			
00000 TOTAL ERRORS					

CHECK	5300	CONV	6FB7	EXEC	538B	FLAG	6FCE	LDHLA	6FA7
LDHLA	6FAE	LOWER	6FC8	PLACIT	6F72	RTNE0	6F9B	RTNE1	6F20
RTNE2	6F2C	RTNE3	6F33	RTNE4	6F38	RTNE5	6F3F	RTNE6	6FAA
RTNE7	6F52	RTNE8	6F8D	RTNE9	6F93	START	5389		

## MULTIDOS 1.7 CHANGES

Information provided by Vern Hester

A new version of MULTIDOS (version 1.7) has recently appeared on the scene. Since many NORTHERN BYTES readers use the previous version of MULTIDOS (version 1.6), I asked MULTIDOS author Vern Hester to provide our readers with a list of the differences between the two versions. Listed here are the changes that were made to version 1.6 to come up with the new version 1.7!

1. Provisions for 8" drive support and HARD disk support, also has been modified to handle floppies in two headed drives as one or two volumes (most other DOSes handle double-sided drives as a single volume, MULTIDOS 1.7 is now compatible with this format).

2. Drives can be configured independently logical and physical.

3. Initialization code reads hardware clocks.
4. BUILD will overwrite as well as append to a "DO" file.
5. Wildcard in DIR and CAT/CMD.
6. Margin added to FORMS.
7. Print directly from DOS command mode.
8. ROUTE an output device to a disk file.
9. Five additional single letter commands in BASIC.
10. BASIC program RAM resident unpacker.
11. Renumber a block of text to a new location.
12. Send control codes to printer from Minidos (MOD I/III).
13. ZAP enhanced to copy sectors, verify sectors, and format a single track.
14. VFU enhanced to permit retries on a function error. (i.e. Write protected diskette)
15. CAT enhanced to indicate granule allocation. (File map)
16. DEBUG enhanced to move one byte at a time and full arrow movement for modification.
17. DDT/CMD enhanced to slow down a MODEL 4 clock when running in the MODEL III mode.
18. Forms filter (MOD I/III) to trap unwanted codes.
19. PACKER modified to absorb blank lines, and not pack lines with open quotes.
20. Enhanced to read double sided NEWDOS/80 diskettes.
21. FORMAT/CMD and BACKUP/CMD made more user friendly by informing user of write protected media instead of exiting.
22. VFU/CMD modified to remove a partial file when disk space is full.
23. Minidos enhanced to turn DEBUG on.

Now here are some specifics of the DOS enhancements:

1. a) Resident module and support overlays have been modified to handle double-sided diskettes as one or two volume.
- b) Drive control table (DCT) has been expanded to ease the incorporation of Hard disk interface, and 8" drive interface.
- c) To make room for the DCT, 'FORMS' has been removed and placed in the command file PRT/CMD. The SPOOLER and library command FORMS require PRT/CMD to be loaded. (MOD I/III)
- d) The initialization code has been modified to read standard hardware clocks (TRSWATCH, NEWCLOCK, etc), and load this information into the software clock. No further reading is performed by MULTIDOS after initialization.

2. System files have been renamed to their primary function and extensions have been changed to DOL (Dos OverLay). BASIC overlays have the extensions changed to BOL (Basic OverLay)

Version 1.6	Version 1.7
DIR/SYS	DIR/SYS
DOS/SYS	SYSRES/SYS
DOS0/SYS	Allocate/DOL
DOS1/SYS	Command/DOL
DOS2/SYS	Open/DOL
DOS3/SYS	Close/DOL
DOS4/SYS	Error/DOL
DOS5/SYS	Debug/DOL
DOS6/SYS	Library1/EXT
DOS7/SYS	Minidos/DOL
DOS8/SYS	Library2/EXT
DOS9/SYS	HELP/DOL or HELP/CMD

#### Library Commands (Functional changes)

The following DOS library commands have NOT been changed: Append, Attrib, Auto, Blink, Boot, Clear, Clock, CIs, Date, Ddam (command exists on MODEL I only), Dead, Device, Do, Dump, Free, Kill, Lib, Link, Load, Prot, Rename, Restor, Setcom (command exists on MODEL III only), Time, Topmem, Verify.

The following DOS library commands have been ELIMINATED: Break, Hash, Help (Now HELP/CMD).

The following DOS library commands have been CHANGED:

Build - Changed to overwrite file unless the A parameter is added.

BUILD filespec (A)

Clrdsk - Changed to place 6DB6 pattern on double density sectors or E5E5 pattern on single density sectors.

Config - Changed to write only if "X" parameter specified.

CONFIG [[:Id]] (m,ST=st,SI=si,V=v,wp/we,P=p,n)]<ENTER>  
 d = logical drive number, 0 to 7  
 m = 5 for 5 1/4" floppy, 8 for 8" floppy, or H for hard disk.  
 st = 6, 12, 20, 30, (track to track step rate -  
 MODEL I 30 = 40 ms.)

si = 1, or 2 for floppies (number of sides)  
 v = 1, or 2 for floppies (volume)  
 wp = write protect  
 we = write enable  
 p = physical drive, 0 to 7  
 n = nil (logical drive not in system)

Debug - Automatic modes E, N & U are cancelled via <SHIFT><SPACE> vs <SPACE> only. Modification mode enhanced to use all four arrow keys. Page display can now be on exact byte with 8 movements using the shift key and all four arrow keys.

Dir - Added wild card (modified CP/M format).

? = character position can be anything.

\* = balance of "field" can be anything.

DIR ?K?????/\* same as DIR ?K\*/\*

this will display files with "K" in second position of filename "field" and anything in extension "field".

DIR \*/CMD display files which end with "/CMD".

DIR \*/ display files without an extension.

DIR ???/\* display files with a max of 3 characters in filename.

Forms - M parameter added for left margin, S parameter dropped. (Requires PRT/CMD loaded - MOD I/III)

Keybrd - reduced to only effect whether case will be in upper or upper/lower case during power-up, and cursor character. (MOD I/III)

List - Modified to convert graphic character to periods unless the G parameter is added.

LIST Picture (G)<ENTER>

D parameter added to perform a sector dump.

LIST DIR/SYS (D)<ENTER>

Patch - T option added.

Patch file (Rec=nn) T= t1,t2,t3>b1,b2,b3<ENTER>

T=target. t1 t2 t3 = target bytes which MUST be found in order to change to b1 b2 b3

Patch DOS/SYS (REC=9) T= 32;194;68>32;194;64

Print - Modified to send string to printer if one double quote is immediately after the "T" in PRINT

PRINT"TEST" lprints TEST

PRINT" TEST" lprints TEST

PRINT""TEST" lprints "TEST"

Route - Enhanced route in output device to a disk file.

Route PR to SCREEN/TXT

Skip - Modified to skip on logical drive zero.

The following DOS library commands have been ADDED:

Reset - Resets devices and TOPMEM to the power-up/re-boot status.

Screen - Dump screen contents to printer.

#### SUPERBASIC

1. BASIC changes include the implementation of addition single letter commands: A for AUTO, L for LOAD, K for KILL, S for SAVE, and I for AUTO (current line+1), 1 (if the current line is 600, I <ENTER> performs an AUTO 601,1 but I300<ENTER> does not perform an AUTO 301,1.

2. a) CMD "O" to open an additional file buffer, has been eliminated

b) CMD"U" to remove remarks has been changed to CMD"X"

c) CMD"X" to transfer the Disk BASIC program to LEVEL II BASIC has been changed to CMD"W"

d) CMD"U" now performs an Unpack. The unpacker breaks each line down to as many single statement lines as possible, inserts spaces around each key word, and rennumbers the program 10, 20, 30, etc.

3. RENUM/SYS renamed RENUM/BOL will move a block of lines to another location provided the new area has room for the new increment.

Syntax: new line, increment, start line, end line

Note: Previous RENUM/SYS would not renumber the end line.

4. BASIC system files:

Version 1.6	Version 1.7
CREF/SYS	CREF/BOL
EDIT/SYS	EDIT/BOL
ERROR/SYS	ERROR/BOL
PACK/SYS	PACK/BOL was in MOD III as a file.
RENUM/SYS	RENUM/BOL
	UNPACK/BOL
	UTIL/BOL

4. UTIL/BOL has been created to keep all of BASIC extended functions in "/BOL" files. This removed the SORT function, CMD"C", M direct command and N direct command from DOS7/SYS; removed the direct command F, and CMD"U" (now CMD"X") from DOS4/SYS; and removed the CMD"UUUUU" initialization routine from DOS0/SYS.

#### REWRITE OF NEWDOS/80 DISK BASIC AMPERSAND (&) ROUTINE by Gil Spencer VK2JK

I never think in octal. It's hard enough to work in binary, hex, and decimal. It always seemed to me that the default for the '&' function should be hex, not octal. I finally dug out the source code (from Apparat's Disk BASIC) which I found in SYS20/SYS. My rewrite fits within the required extra space. Although a quantity of bytes are changed, this is because the code is "re-arranged" more than because it is "re-written".

First, here is the Disk BASIC (&) routine found in Apparat's NEWDOS/80 version 2.0 - specifically SYS20/SYS, addresses 54C5H-5503H. If you are using SUPERZAP, address 54C5H is found at FRS 2, byte D1H and address 5503H is at FRS 3, byte 13H. Note that the four bytes at FRS 3, bytes 06H-09H (which are 01 00 FA 54) are loader codes and must NOT be changed.

54C5	00100	ORG	54C5H
54C5 D7	00110	RST	10H
54C6 4F	00120	LD	C,A
54C7 110000	00130	LD	DE,0000H
54CA 79	00140 Q54CAH	LD	A,C
54CB FE48	00150	CP	48H
54CD 2022	00160	JR	NZ,Q54F1H
54CF D7	00170	RST	10H
54D0 EB	00180	EX	DE,HL
54D1 D630	00190	SUB	30H
54D3 FE0A	00200	CP	0AH
54D5 3808	00210	JR	C,Q54DFH
54D7 D611	00220	SUB	11H
54D9 FE06	00230	CP	06H
54DB 3022	00240	JR	NC,Q54FFH
54DD C60A	00250	ADD	A,0AH
54DF 29	00260 Q54DFH	ADD	HL,HL
54E0 3807	00270	JR	C,Q54E9H
54E2 29	00280 Q54E2H	ADD	HL,HL
54E3 3804	00290	JR	C,Q54E9H
54E5 29	00300	ADD	HL,HL
54E6 3801	00310	JR	C,Q54E9H
54E8 29	00320	ADD	HL,HL
54E9 DAB207	00330 Q54E9H	JP	C,07B2H
54EC 85	00340	ADD	A,L
54ED 6F	00350	LD	L,A
54EE EB	00360	EX	DE,HL
54EF 18D9	00370	JR	Q54CAH
54F1 0E4F	00380 Q54F1H	LD	C,4FH
54F3 B9	00390	CP	C
54F4 2801	00400	JR	Z,Q54F7H
54F6 2B	00410	DEC	HL
54F7 D7	00420 Q54F7H	RST	10H
54F8 EB	00430	EX	DE,HL
54F9 D630	00440	SUB	30H
54FB FE08	00450	CP	08H
54FD 38E3	00460	JR	C,Q54E2H
54FF CD9A0A	00470 Q54FFH	CALL	0A9AH
5502 EB	00480	EX	DE,HL
5503 C9	00490	RET	

This is the rewrite of the Disk BASIC ampersand (&) routine. Now the octal argument must be specified by '&O'. Hex argument may be specified by '&H'. No suffix (i.e. '&') now defaults to hex rather than octal.

54C5	00100	ORG	54C5H
54C5 D7	00110	RST	10H
54C6 4F	00120	LD	C,A
54C7 110000	00130	LD	DE,0000H
54CA 79	00140 Q54CAH	LD	A,C
54CB FE48	00150	CP	48H
54CD 2022	00160	JR	NZ,Q54F1H
54CF D7	00170	RST	10H
54D0 EB	00180	EX	DE,HL
54D1 D630	00190	SUB	30H
54D3 FE0A	00200	CP	0AH
54D5 3808	00210	JR	C,Q54DFH
54D7 D611	00220	SUB	11H

54D9 FE06	00230	CP	06H
54DB 3022	00240	JR	NC,Q54FFH
54DD C60A	00250	ADD	A,0AH
54DF 29	00260 Q54DFH	ADD	HL,HL
54E0 3807	00270	JR	C,Q54E9H
54E2 29	00280 Q54E2H	ADD	HL,HL
54E3 3804	00290	JR	C,Q54E9H
54E5 29	00300	ADD	HL,HL
54E6 3801	00310	JR	C,Q54E9H
54E8 29	00320	ADD	HL,HL
54E9 DAB207	00330 Q54E9H	JP	C,07B2H
54EC 85	00340	ADD	A,L
54ED 6F	00350	LD	L,A
54EE EB	00360	EX	DE,HL
54EF 18D9	00370	JR	Q54CAH
54F1 0E4F	00380 Q54F1H	LD	C,4FH
54F3 B9	00390	CP	C
54F4 2801	00400	JR	Z,Q54F7H
54F6 2B	00410	DEC	HL
54F7 D7	00420 Q54F7H	RST	10H
54F8 EB	00430	EX	DE,HL
54F9 D630	00440	SUB	30H
54FB FE08	00450	CP	08H
54FD 38E3	00460	JR	C,Q54E2H
54FF CD9A0A	00470 Q54FFH	CALL	0A9AH
5502 EB	00480	EX	DE,HL
5503 C9	00490	RET	

#### CREATE A SELF-BOOTING DISK USING LDOS by Gary Bryce

[Reprinted from SYDTRUG NEWS, P.O. Box 297, Padstow, New South Wales 2211, AUSTRALIA]

This should be the last word on this subject. Back in the August issue of the [SYDTRUG] newsletter I detailed a method of creating a self booting disk using NEWDOS/80 on the Model I (Leon Yates followed this with the details for the Model III in December) [See NORTHERN BYTES Volume 5, Number 4, page 11; Volume 6, Number 1, page 3, plus THE EXTERMINATOR column in Volume 5, Number 6, page 2, and THE EXTERMINATOR in this issue for these past articles], there was quite a bit of zapping and general playing around to do it, but it did work. One evening recently I was pondering on how to do the same thing with an LDOS disk when I realised how much simpler it would be using an LDOS formatted diskette.

Why is it simpler? Well what we are doing is substituting the file to load for SYS0 and using the normal SYS0/SYS loader in BOOT/SYS to load and execute our file. NEWDOS/80 doesn't look at the directory to load SYS0/SYS, it depends on the Track and Sector numbers at the start of SYS0 being in the BOOT/SYS file, after loading SYS0/SYS it performs a check to ensure a correct load. Therefore I had to ensure that the correct Track & Sector numbers for the file are correct and also bypass the check. LDOS accesses the directory to get the position of SYS0/SYS on the disk, so if we put the file in the directory slot normally used by SYS0/SYS the Boot loader will load our file and execute it! How do we do it? As I said, it couldn't be simpler - copy the file to SYS0/SYS!

The procedure is very easy, just follow the steps listed below and you can't go wrong, and it works for the Model I and Model III.

1. Format a diskette using LDOS.
2. Create the SYS0/SYS file on the diskette using the following command:

BACKUP SYS0/SYS:d

where d is the destination drive number.

3. Copy the required file to the diskette:

COPY FILENAME/CMD:s SYS0/SYS.RS0LT0FF:d

where s is the source drive and d is the destination drive.

4. Optionally you may then rename SYS0/SYS back to the original filename by:

RENAME SYS0/SYS.RS0LT0FF:d TO FILENAME/CMD

And there you have it, no zapping or calcs required. Why do it at all? Most of us use a directory indexing program of some description; conventional self booting diskettes don't have a directory and therefore can't be read and must be added manually. Using this method the diskette has a conventional directory and can be read by the indexing program.



# MODULA-2!

## NOW AVAILABLE

The Alternate Source Information Outlet is now the North American distributor for the Hochstrasser Computing Modula-2 System for Z80 CP/M! If you thought that Modula-2 wasn't possible on an 8-bit machine, keep reading! The programmer goals were to create a full Modula-2 compiler for an 8-bit machine that generates a reasonably small amount of code in a reasonably small amount of time leading to reasonably short execution times. These goals are now realities!

## WHAT MAKES MODULA-2 SO GREAT?

In several PASCAL dialects, there exists possibilities to split programs apart into different "modules". These modules are compiled as if they were complete programs just lacking the main program. You can freely use procedures declared in one such "module" in others. All you have to do is to declare how this procedure looks and indicate that it is "external". It is a tool that is indispensable for the successful mastery of large programming projects. It is especially helpful for projects that are carried out by groups of programmers. In programmer terms, Modula-2 takes the concept of "local" and "global" variables to new dimensions. The programmer has complete control.

## THE DOCUMENTATION

Manual Release 3-28-85/pwh -- almost 300 pages -- is very complete and well indexed (nine pages of carefully thought out subjects and subdivisions). It is presumed that the user has a some familiarity with PASCAL. Please note that the current documentation has been updated since an older version of this product was critiqued in an issue of BYTE. For persons who get squeemish at the thought of spending money, we have constructed a "Modula Sales Kit" which includes the complete Table of Contents and other information from the manual. One of these is free upon request.

## POWER IS WHAT YOU GET

The Modula-2 Compiler Package for Z80 CP/M includes a linker, a reference lister, a converter, a system configuration package, two libraries (detailed below) and three test programs. Complete step-by-step instructions are included for assembling the test programs. The System generates fast, ROMable, reentrant Z80 native code. Assembly language integration is supported, as well as assembly language compiler output.

## SOFTWARE SUPPORT

Included with this version of Modula-2 is a complete set of source code library routines including **TERMINAL**, **SEQIO**, **TEXTS**, **REALTEXTS**, **INOUT**, **REALINOUT**, **MATHLIB**, **SYSTEM**, **ASCII**, **CHAINING**, **CMDLIN**, **CONTROLS**, **STRINGS**, **LONGSETS**, **CONVERSIONS**, **CONVERTREAL**, **FILENAME**, **FILESYS**, **FILES**, **MOVES** and **OPSYS**. The new documentation includes a **START-UP GUIDE**, **INTRODUCTION TO MODULA-2**, **IMPLEMENTATION GUIDE**, **ADVANCED PROGRAMMING GUIDE** and **APPENDICES** that detail error messages, object code format considerations, reserved words and symbols, the ASCII character set, language definition, a bibliography and 9 pages of index. A special section is devoted to Programming With Better Efficiency. The documentation is filled with small sample Modula-2 programs.

## SYSTEM CONSIDERATIONS

Please note that this is a large system. The compiler itself uses about 170k of disk space. It is desirable to have at least two drives holding 350 Kilobytes of disk storage each to work comfortably with the system. A single double-sided eighty (holding 700k) would be ideal. Unless otherwise requested, the Modula-2 Compiler for Z80 CP/M will be shipped using the Montezuma Micro 40-track, single-side format. The compiler is only tested under CP/M 2.2. Eight inch CP/M formats are also available upon request.

## FOR NEW MODULA-2 PROGRAMMERS

Folks near the mid-Michigan area: This summer we are planning at least one seminar detailing the use of Modula-2. We have a special package price on the seminar, which includes the price of the compiler and complete documentation, along with **PERSONAL INSTRUCTION** and enough diskettes to make a backup of your software and for working/scratch purposes. The price of the Modula-2 Compiler Package for CP/M, including the seminar and all mentioned above is \$200. This price requires that you bring **YOUR OWN CP/M computer**. The price for the seminar with **OUR computer** is \$300. The price for the software without the seminar is \$165. The seminar without the software (you provide your own Modula-2 compiler and computer) is \$59.95 if you make your reservations by August 1st, \$99.95 after that date. The 8-hour seminar will take place in Lansing on Saturday, August 10th, and include lunch. Advanced seminars are planned and will depend on the success of this seminar, naturally.

## WHERE TO GET IT!

THE  
**ALTERNATE**  
SOURCE

TAS I/O  
704 N. Pennsylvania  
Lansing, MI 48906  
(517) 482-8270

# NORTHERN BYTES

## Subscription Information

Northern Bytes is edited by Jack Decker and published on an irregular basis by The Alternate Source Information Outlet. Back issues are available starting with Volume 5, Number 1. Issues prior to that are not available. Some of the most valuable articles from earlier issues may be reprinted in future issues of Northern Bytes. Currently there are eight back issues available for Volume 5, as well as all issues from Volume 6. All back issues are \$2 each.

It is very easy to be placed on the Northern Bytes REGULAR list. Simply place your address, Visa or MasterCard number and expiration date on file with us. We will start with the issue you request. We do not bill you for ANY ISSUE until that issue has been mailed. This way, we can continue to offer you top quality information with absolutely no risk to you. There's no question of what to

do about unfulfilled issues if we decide to quit publishing. Unless otherwise requested, we presume your subscription will extend through the month of your expiration date.

Don't have a charge card, huh? We understand the myriad of reasons for not having them and we feel that a "To-Be-Invoiced" policy could help increase the demand for Northern Bytes. If you'll do it for us, we'll do it for you. Would you like to be placed on a regular list TO BE BILLED for each issue? You could then send a check for the issues as they are mailed. If you didn't send a check, we would presume that your interest has died and discontinue your subscription. The only requirement for getting onto the list is to pay for the first issue up front; the next will be mailed automatically. If you request, you are insured that you will receive top of the line TRS-80 information as it is released. Ask to be placed on the NO RISK "To-Be-Invoiced Northern Bytes List".

Call or write, but SIGN UP TODAY!  
**The Alternate Source Information Outlet**  
**704 North Pennsylvania Avenue**  
**Lansing, MI 48906**  
**(517) 482-8270**

## NORTHERN BYTES

c/o Jack Decker  
1804 West 18th Street  
Lot # 155  
South Ste. Marie, Michigan 49783  
MCI Mail Address: 102-7413  
Telex: 6501027413  
(Answerback: 6501027413 MCI)

Bulk Mail  
U.S. Postage Paid  
Permit #815  
Lansing, MI

**POSTMASTER: If undeliverable return to:**

The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan 48906

**To:**